A Domain Decomposition Method for First-order PDEs

by

Lina Hemmingsson

Report No. 161/1994

UPPSALA UNIVERSITET
Institutionen för
teknisk databehandling

UPPSALA UNIVERSITY
Department of
Scientific Computing

A Domain Decomposition Method for First-order PDEs[1]

Lina Hemmingsson

June, 1994

[1] This work was supported by the Swedish National Board for Industrial and Technical Development, NUTEK, under contract No. 9303806

Abstract

In this report a non-overlapping domain decomposition method to solve first-order, time-dependent partial differential equations is developed. The time-discretization used is implicit, which gives a large system of equations to solve for, each time-step. Preconditioners with a fast inversion based on a fast modified sine transform are defined. Theoretical analysis of the method is presented, indicating that the ratio in the grid might be crucial for the convergence. Finally numerical results from a parallel implementation on an Intel iPSC/2 and an Intel Paragon are presented, showing very nice properties. For one version of the domain decomposition method, a nonuniform decomposition of the domain lead to very good results.

Contents

1 Introduction

We consider systems of time-dependent, first-order partial differential equations in two space-dimensions

$$\begin{cases} \dfrac{\partial u}{\partial t} + B_1(x_1,x_2)\dfrac{\partial u}{\partial x_1} + B_2(x_1,x_2)\dfrac{\partial u}{\partial x_2} = g \quad , \quad x \in \Omega \ , t \ \geq 0 \\ +(\text{boundary conditions}) \\ +(\text{initial conditions}). \end{cases} \tag{1}$$

Here B_1 and B_2 are $n_c \times n_c$-matrices, and u and g are n_c-vectors.

Systems like (1), are often discretized explicitly in time. In problems with different time-scales, where we are only interested in the slowly varying part of the solution, the restriction on the time-step due to the CFL-criterion for the fast oscillations, becomes unrealistic in some applications. If we are only interested in the slowly varying part of the solution, we can discretize implicitly in time with a large time-step, and still obtain an accurate solution [8], [9], [20]. Another interesting case is when the geometry is complicated such that the smallest cell in the space-grid, restricts the time-step severely in an explicit time-discretization. Then an implicit discretization in time is preferable. This gives a large system of equations

$$Au = b \tag{2}$$

to solve for each time-step.

The time-step used is large compared to the smallest space-step. Hence A is strongly non-diagonally dominant. Moreover it is non-symmetric. Holmgren and Otto have investigated a number of iterative methods and block- or semi-circulant preconditioning matrices for (2), showing very good results, [14], [15]. In [11], preconditioners based on a Toeplitz structure are examined, showing similar results.

Large scale problems will still be very time-expensive, and the necessity to use a more efficient method becomes obvious. Over the past several years much effort has been focused on investigating domain decomposition methods on multiprocessor computers, [3], [4], [5], [16]. In these techniques, the domain is divided in several subdomains. Each processor is given one or more subdomains and they can work in parallel with communication only on the interfaces. In this report a non-overlapping domain decomposition method to solve Equation (2) is developed and examined.

2 The domain decomposition setting

2.1 Introduction

The first known domain decomposition algorithm was introduced by Schwarz in 1869, [19]. The idea was to use known solvers for elliptic equations on simple geometries such as circles and rectangles, to solve the equation on a more complex domain. This is known as the Schwartz alternating procedure.

In this report we study the case with non-overlapping subdomains. In Figure 1 the original domain is divided in four rectangular subdomains. The idea is to reduce the original system to a smaller system for the unknowns on the interior boundaries (dashed lines). This smaller system is called the Schur-Complement system. The solution of this system includes the solution of the different subdomain systems. These subdomain systems can be solved using known solution methods for rectangles. Once the Schur-Complement system is solved, each subdomain system can be solved seperately with no communication. This can be exploited using a parallel computer. The inherent parallelism in this solution method leads to the suggestion that one might use this method even on more regular domains. Implementing the method on a parallel computer could lead to an considerable gain in time. It also gives the possibility to solve larger problems as each processor only stores the unknowns of its awarded subdomains.

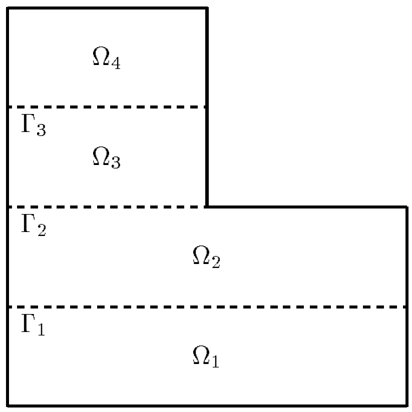

Figure 1: The domain decomposed in four subdomains.

Now study some arbitrary domain that we can divide in p rectangular subdomains. We will restrict our formulation to the case with interior boundaries parallel to the x_1-axis. Denote each subdomain Ω_q and the interior boundaries

Γ_q. Also denote $\bigcup_{q=1}^{p} \Omega_q = \Omega$ and $\bigcup_{q=1}^{(p-1)} \Gamma_q = \Gamma$. The unknowns in Ω_q are denoted $u_1^{(q)}$ and the unknowns on Γ_q $u_0^{(q)}$.

If we arrange the system such that we get the unknowns on Γ first and those of the subdomains last, we obtain a system of equations for the solution vector $(u_0, u_1)^T$, which we express in block form as

$$\mathcal{A} \begin{pmatrix} u_0 \\ u_1 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \end{pmatrix}, \tag{3}$$

where

$$\mathcal{A} = \begin{pmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{pmatrix},$$

or more explicitly expressed in terms of the different subdomains

$$\mathcal{A} = \left(\begin{array}{ccc|cccc} A_{00}^{(1)} & & & A_{01}^{(11)} & A_{01}^{(12)} & & \\ & \ddots & & & \ddots & \ddots & \\ & & A_{00}^{(p-1)} & & & A_{01}^{((p-1)(p-1))} & A_{01}^{((p-1)p)} \\ \hline A_{10}^{(11)} & & & A_{11}^{(1)} & & & \\ A_{10}^{(21)} & \ddots & & & A_{11}^{(2)} & & \\ & \ddots & A_{10}^{((p-1)(p-1))} & & & \ddots & \\ & & A_{10}^{(p(p-1))} & & & & A_{11}^{(p)} \end{array} \right).$$

2.2 Schur-Complement Methods (SCM)

In a Schur-Complement Method, [3], [4], [5], [16], Equation (3) is reduced to a system for the unknowns on Γ

$$Cu_0 = g, \tag{4}$$

where

$$C = A_{00} - A_{01} A_{11}^{-1} A_{10}, \tag{5}$$

and

$$g = b_0 - A_{01} A_{11}^{-1} b_1.$$

Equation (4) is solved for, and u_0 is inserted into (3). The remaining unknowns $u_1^{(q)}$ can then be computed locally in Ω_q. This computation is the same as solving the differential equation on Ω with Dirichlet boundary conditions u_0 on Γ. The inherent parallelism in the algorithm can be exploited using a computer with a parallel architecture.

The algorithm to solve our problem with SCM we now state as follows

---

Algorithm SCM

1. Solve $A_{11}w = b_1$
2. Solve $Cu_0 = b_0 - A_{01}w$
3. Solve $A_{11}v = A_{10}u_0$
4. $u_1 = w - v$

Step 1. and 3.,

$$\text{Solve } A_{11}x_1 = y_1 \tag{6}$$

are local subdomain solves,

$$\text{Solve } A_{11}^{(q)}x_1^{(q)} = y_1^{(q)} \quad , \quad q = 1, \ldots, p, \tag{7}$$

that can be effectuated totally in parallel with no communication. Similarly step 4. is a local operation, while step 2. demands the solution of a system that is spread over the processors.

A direct solution method of Equation (4) (step 2. in **Algorithm SCM**), implies that we first have to evaluate C. This requires the solution of $2n$ subdomain systems in each subdomain, which is quite demanding in time when it comes to large scale problems. Here n is the number of unknowns in space-direction x_1. LU-factorization of the evaluated C requires $\mathcal{O}(n^3n_c^3)$ operations, and the back-substitution $\mathcal{O}(n^2n_c^2)$ operations for each time-step. This is also an expensive process for large n, which leads us to investigate more efficient methods.

Classical iterative methods such as SOR and Gauss-Seidel do not work in our case, since these methods require diagonally dominant coefficient matrices. More modern iterative methods are Conjugate Gradient Methods. These methods require symmetric systems which means that they will not do either. There is however a class of Generalized Conjugate Gradient Methods which work for nonsymmetric systems as well. In this application we will employ an iterative method called PGCR(ℓ) - **Preconditioned Generalized Conjugate Residuals** with restarting length ℓ, [6]. From [11], we conclude that this iterative method with an efficient preconditioner also will be a good solution method for the subdomain solves, step 1. and 3. in **Algorithm SCM**. Hence this is the method used.

The SCM-method might be somewhat inconvenient when we are using an iterative method to solve the Schur-Complement system. First we have to decide how accurate to carry out step 1. and 2. in **Algorithm SCM**. Then we have to carry out step 3., such that the convergence criterion used is fulfilled. Thus the accuracy in w and u_0 will affect the total number of iterations needed to obtain the accuracy called for, and consequently the performance of the method. Furthermore, we must deal with the subdomain solves in the matrix-vector-multiplications Cx that we perform each iteration-step. Studying C we see that each such multiplication requires local subdomain solves in each subdomain. Hence a stop-criterion is needed for these inner iterations, which is a parameter that also affects the performance. In Section 9.2.1 we describe how to partly circumvent these problems.

2.3 Partitioned Matrix Methods (PMM)

The difficulties with SCM discussed in the previous section leads to the suggestion that one should iterate on the partitioned system (3), i.e., one iterates on Γ and Ω simultaneously. A direct solution method of Equation (3) causes much fill-in which is not desirable. Thus an iterative method is employed here as well as in the SCM-case. With the same discussion as in the previous section, we conclude that the PGCR-method will suit our purposes.

3 Iterative methods

Consider a general system of equations

$$Bx = y. \tag{8}$$

PGCR is a minimal residual iteration [7], which in step k fulfills

$$\|r_k\|_2 = \min_{p_k \in \mathcal{P}_k,\, p_k(0)=1} \|p_k(\hat{B}^{-1}B)r_0\|_2,$$

where $\mathcal{P}_k$ is the set of all polynomials of degree k, $\hat{B}$ is the left preconditioner, $r_k \equiv \hat{B}^{-1}(Bx_k - y)$, and x_k is the approximation of x obtained in iteration k. If $\hat{B}^{-1}B$ is diagonalizable we obtain

$$\frac{\|r_k\|_2}{\|r_0\|_2} \leq \mathrm{cond}_2(W_{\hat{B}^{-1}B}) \min_{(p_k \in \mathcal{P}_k,\, p_k(0)=1)} \max_{1 \leq \ell \leq \eta} |p_k(\lambda_\ell)| \equiv \mathrm{cond}_2(W_{\hat{B}^{-1}B}) \cdot \varepsilon_k, \tag{9}$$

where $W_{\hat{B}^{-1}B}$ is the eigenvector-matrix and λ_ℓ are the eigenvalues of $\hat{B}^{-1}B$. We also define the asymptotic convergence factor ρ as

$$\rho \equiv \lim_{k\to\infty} \varepsilon^{1/k}. \tag{10}$$

From Equation (9) we conclude that we will have finite termination in η iterations, where η is the number of distinct eigenvalues to the preconditioned system. Moreover, if we can precondition our system such that the eigenvalues of $\hat{B}^{-1}B$ are contained in μ dense clusters, we have a good approximation to the solution in μ iterations. Clustering of the eigenvalues may be even more important than a condition number improvement, [1], [2], and [21]. Experience from [11] shows that even though we do not know if the preconditioned system is diagonalizable, relation (9) seems to be valid. In Section 6, preconditioners to (3), (4), and (6) are presented, that yield highly clustered spectra. We also show in Section 7.1.1, that for the SCM-case, it might not be necessary to precondition the Schur-Complement system.

4 Equivalence between SCM and PMM

Keyes and Gropp have shown that SCM and PMM are equivalent in a certain sense, when we are dealing with symmetric and positive definite matrices, [16]. We will here show that this holds, when we apply PGCR(ℓ), even though the matrices are nonsymmetric and indefinite.

Theorem 4.1 *PGCR(ℓ) on (4) with initial iterate $u_0^{(0)}$ and preconditioner $\hat{C}$ is equivalent to PGCR(ℓ) on (3) with initial iterate*

$$\begin{pmatrix} u_0 \\ u_1 \end{pmatrix}^{(0)} = \begin{pmatrix} u_0^{(0)} \\ A_{11}^{-1}(b_1 - A_{10}u_0^{(0)}) \end{pmatrix},$$

and preconditioner

$$\hat{\mathcal{A}} = \begin{pmatrix} \hat{C} + A_{01}A_{11}^{-1}A_{10} & A_{01} \\ A_{10} & A_{11} \end{pmatrix}, \tag{11}$$

in the sense that for all $k \geq 0$

$$\begin{pmatrix} u_0 \\ u_1 \end{pmatrix}^{(k)} = \begin{pmatrix} u_0^{(k)} \\ A_{11}^{-1}(b_1 - A_{10}u_0^{(k)}) \end{pmatrix}.$$

Proof:
Any non-singular matrix can be factorized into an upper-triangular matrix U and a lower-triangular matrix L. That means that $\hat{C}$ can be factorized as $\hat{C} = UL$. Hence $\hat{\mathcal{A}}$ can be factorized as $\hat{\mathcal{A}} = \mathcal{U}\mathcal{L}$ where

$$\mathcal{U} = \begin{pmatrix} U & A_{01} \\ 0 & A_{11} \end{pmatrix},$$

and

$$\mathcal{L} = \begin{pmatrix} L & 0 \\ A_{11}^{-1}A_{10} & I \end{pmatrix}.$$

PGCR(ℓ) on (4) is equivalent to GCR(ℓ) on $\check{C}\check{u}_0 = \check{g}$, with initial iterate $\check{u}_0^{(0)} = 0$, where $\check{C} \equiv U^{-1}CL^{-1}$, $\check{u}_0 \equiv L(u_0 - u_0^{(0)})$, and $\check{g} \equiv U^{-1}(g - Cu_0^{(0)})$. In the same way PGCR($\ell$) on (3) is equivalent to GCR(ℓ) on $\check{\mathcal{A}}\check{u} = \check{b}$, with initial iterate

$$\check{u}^{(0)} = \begin{pmatrix} \check{u}_0^{(0)} \\ 0 \end{pmatrix}.$$

Here $\check{\mathcal{A}} = \mathcal{U}^{-1}\mathcal{A}\mathcal{L}^{-1}$,

$$\check{u} \equiv \mathcal{L}\left(\begin{pmatrix} u_0 \\ u_1 \end{pmatrix} - \begin{pmatrix} u_0^0 \\ A_{11}^{-1}(b_1 - A_{10}u_0^0 \end{pmatrix}\right),$$

and

$$\check{b} \equiv \mathcal{U}^{-1}\left(\begin{pmatrix} b_0 \\ b_1 \end{pmatrix} - \begin{pmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{pmatrix}\begin{pmatrix} u_0^{(0)} \\ A_{11}^{-1}(b_1 - A_{10}u_0^{(0)} \end{pmatrix}\right).$$

But note that

$$\begin{aligned}
\check{\mathcal{A}} &= \begin{pmatrix} U & A_{01} \\ 0 & A_{11} \end{pmatrix}^{-1} \begin{pmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{pmatrix} \begin{pmatrix} L & 0 \\ A_{11}^{-1}A_{10} & I \end{pmatrix}^{-1} \\
&= \begin{pmatrix} U^{-1} & -U^{-1}A_{01}A_{11}^{-1} \\ 0 & A_{11}^{-1} \end{pmatrix} \begin{pmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{pmatrix} \begin{pmatrix} L^{-1} & 0 \\ -A_{11}^{-1}A_{10}L^{-1} & I \end{pmatrix} \\
&= \begin{pmatrix} U^{-1}(A_{00} - A_{01}A_{11}^{-1}A_{10})L^{-1} & 0 \\ 0 & I \end{pmatrix} = \\
&= \begin{pmatrix} U^{-1}CL^{-1} & 0 \\ 0 & I \end{pmatrix} = \begin{pmatrix} \check{C} & 0 \\ 0 & I \end{pmatrix}.
\end{aligned}$$

Similarly

$$\begin{aligned}
\check{b} &= \begin{pmatrix} U & A_{01} \\ 0 & A_{11} \end{pmatrix}^{-1} \left(\begin{pmatrix} b_0 \\ b_1 \end{pmatrix} - \begin{pmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{pmatrix} \begin{pmatrix} u_0^{(0)} \\ A_{11}^{-1}(b_1 - A_{10}u_0^{(0)}) \end{pmatrix}\right) \\
&= \begin{pmatrix} U^{-1} & -U^{-1}A_{01}A_{11}^{-1} \\ 0 & A_{11}^{-1} \end{pmatrix} \begin{pmatrix} b_0 - A_{00}u_0^{(0)} - A_{01}A_{11}^{-1}(b_1 - A_{10}u_0^{(0)}) \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} U^{-1}(b_0 - A_{01}A_{11}^{-1}b_1 - (A_{00} - A_{01}A_{11}^{-1}A_{10})u_0^{(0)}) \\ 0 \end{pmatrix} = \\
&= \begin{pmatrix} U^{-1}(g - Cu_0^{(0)}) \\ 0 \end{pmatrix} = \begin{pmatrix} \check{g} \\ 0 \end{pmatrix},
\end{aligned}$$

and

$$
\begin{aligned}
\breve{u} &= \begin{pmatrix} L & 0 \\ A_{11}^{-1}A_{10} & I \end{pmatrix} \begin{pmatrix} u_0 - u_0^{(0)} \\ u_1 - A_{11}^{-1}(f_1 - A_{10}u_0^{(0)}) \end{pmatrix} \\
&= \begin{pmatrix} L(u_0 - u_0^{(0)}) \\ A_{11}^{-1}A_{10}(u_0 - u_0^{(0)}) + u_1 - A_{11}^{-1}(b_1 - A_{10}u_0^{(0)}) \end{pmatrix} \\
&= \begin{pmatrix} L(u_0 - u_0^{(0)}) \\ u_1 - A_{11}^{-1}(b_1 - A_{10}u_0) \end{pmatrix} = \begin{pmatrix} \breve{u}_0 \\ \varphi \end{pmatrix}.
\end{aligned}
$$

Thus, GCR(ℓ)-iteration on $\breve{A}\breve{u} = \breve{b}$ is equal to GCR(ℓ)-iteration on

$$
\begin{pmatrix} \breve{C} & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} \breve{v} \\ \varphi \end{pmatrix} = \begin{pmatrix} \breve{g} \\ 0 \end{pmatrix},
$$

which means that $\varphi^{(k)} \equiv 0$. Hence $u_1^{(k)} = A_{11}^{-1}(b_1 - A_{10}u_0^{(k)})$ and we get

$$
\begin{pmatrix} u_0 \\ u_1 \end{pmatrix}^{(k)} = \begin{pmatrix} u_0^{(k)} \\ A_{11}^{-1}(b_1 - A_{10}u_0^{(k)}) \end{pmatrix}
$$

which proves the theorem. □

From Theorem 4.1 we get a relation between the SCM-iteration and the PMM-iteration. However, the preconditioner defined in (11) yields the exact subdomain solves, which leads to a very time-consuming preconditioner solve. Hence, this approach will not be considered any further.

5 The model problem

5.1 The differential equation

The model problem studied is a scalar two-dimensional equation

$$
\begin{gathered}
\frac{\partial u}{\partial t} + \sigma_1(x_1)\frac{\partial u}{\partial x_1} + \sigma_2(x_2)\frac{\partial u}{\partial x_2} = g, \\
0 < x_1 \leq 1\,,\ 0 < x_2 \leq 1\,,\ t > 0,
\end{gathered} \tag{12}
$$

where $\sigma_1(x_1) > 0$ and $\sigma_2(x_2) > 0$. Equation (12) is well-posed if we prescribe $u(x_1, 0, t)$, $u(0, x_2, t)$ and $u(x_1, x_2, 0)$. We could also consider periodic boundary condition in either space-direction. Note that since we are aiming at solving systems of PDEs, we cannot use any particular features of the scalar equation.

5.2 Discretization

Introduce a uniform grid as

$$\begin{cases} x_{1,k} = kh_1 \quad , \quad k = 1, \ldots, n \\[2ex] x_{2,j} = jh_2 \quad , \quad j = 1, \ldots, m. \end{cases}$$

Let $u_{k,j}$ denote the approximate solution at the point $(x_{1,k}, x_{2,j})$. Equation (12) is discretized in time using the trapezoidal rule with time-step Δt. For the space-derivatives we use centered differences in the interior of the domain and one-sided differences at the outflow boundaries ($k = n$ or $j = m$).

$$\begin{cases} \dfrac{\partial u}{\partial x_1} \approx D_{0,x_1} u_{k,j} & \\ & , \quad k = 1, \ldots, n-1 \text{ and } j = 1, \ldots, m-1 \\ \dfrac{\partial u}{\partial x_2} \approx D_{0,x_2} u_{k,j} & \\[2ex] \dfrac{\partial u}{\partial x_1} \approx D_{-,x_1} u_{n,j} & , \quad j = 1, \ldots, m \\[2ex] \dfrac{\partial u}{\partial x_2} \approx D_{-,x_2} u_{k,m} & , \quad k = 1, \ldots, n \end{cases} \tag{13}$$

The two latter relations will be referred to as the numerical outflow boundary conditions.

To give the equations a simpler appearance we define the following quantities

$$\begin{cases} \kappa_1 = \dfrac{\Delta t}{h_1} \\[2ex] \kappa_2 = \dfrac{\Delta t}{h_2} \\[2ex] \tilde{\kappa}_{1,k} = \sigma_1(x_{1,k})\kappa_1 \quad , \quad k = 1, \ldots, n \\[2ex] \tilde{\kappa}_{2,j} = \sigma_2(x_{2,j})\kappa_2 \quad , \quad j = 1, \ldots, m \end{cases}$$

$$u^\nu = \begin{pmatrix} u^\nu_{1,1} & u^\nu_{2,1} & \ldots & u^\nu_{n,1} & u^\nu_{1,2} & \ldots & u^\nu_{n,m} \end{pmatrix}^T.$$

Rearranging of the equations leads to a system of equations to solve for each time-step

$$Au^{\nu+1} = b. \tag{14}$$

Here b contains known quantities and A consists of blocks D_k and G_k like

$$A = \begin{pmatrix} D_1 & G_1 & & & \\ -G_2 & D_2 & G_2 & & \\ & \ddots & \ddots & \ddots & \\ & & -G_{m-1} & D_{m-1} & G_{m-1} \\ & & & -2G_m & D_m + 2G_m \end{pmatrix},$$

$$D_j = \begin{pmatrix} 4 & \tilde{\kappa}_{1,1} & & & \\ -\tilde{\kappa}_{1,2} & 4 & \tilde{\kappa}_{1,2} & & \\ & \ddots & \ddots & \ddots & \\ & & -\tilde{\kappa}_{1,n-1} & 4 & \tilde{\kappa}_{1,n-1} \\ & & & -2\tilde{\kappa}_{1,n} & 4 + 2\tilde{\kappa}_{1,n} \end{pmatrix} \quad , \quad j = 1, \ldots, m$$

$$G_j = \tilde{\kappa}_{2,j} I_n \quad , \quad j = 1, \ldots, m,$$

where I_n denotes the identity-matrix of order n. The coefficient-matrix A in (14) can also be written

$$A = 4I_{n\times m} + \kappa_1 I_m \otimes A_1 + \kappa_2 A_2 \otimes I_n, \tag{15}$$

where A_1 is a matrix of order n and A_2 a matrix of order m defined by

$$A_d = \begin{pmatrix} 0 & 1 & & & \\ -1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 0 & 1 \\ & & & -2 & 2 \end{pmatrix}.$$

Next we will show that for constant coefficients, the coefficient-matrix $\mathcal{A}$ defined in (3), and the Schur-Complement C defined in (4) are nonsingular. For constant coefficients we define

$$\tilde{\kappa}_d \equiv \sigma_d \kappa_d \quad , \quad d = 1, 2\ .$$

Equation (3) is simply a rearranging of the unknowns from (14), so if we can show that A in (14) and (15) is nonsingular, then $\mathcal{A}$ in (3) will also be nonsingular. In that case we know that a unique solution exists, and hence the Schur-Complement C in (4) will also be nonsingular. In order to prove nonsingularity, we restate a lemma from [17]. Let $\mathcal{E}(a, b)$ denote the closed ellipse centered at the origin with semimajor axis b oriented along the imaginary axis and semiminor axis a. Also let $\mathcal{E}^+(a, b)$ denote the region $\{z | z \in \mathcal{E}(a, b)$ and $\Re e(z) \geq 0\}$.

Lemma 5.1 *The eigenvalues $\lambda_{1,k}$, $k = 1, \ldots, n$ of A_1 and the eigenvalues $\lambda_{2,j}$, $j = 1, \ldots, m$ of A_2 satisfy:*

- $\lambda_{d,k} \neq \lambda_{d,j}$, $k \neq j$
- $\lambda_{1,k} \in \mathcal{E}^+(4n^{-3/4}, 2 + 4n^{-3/2})$
- $\lambda_{2,k} \in \mathcal{E}^+(4m^{-3/4}, 2 + 4m^{-3/2})$

With this lemma we will now prove the nonsingularity of A.

Theorem 5.2 *The coefficient-matrix A defined in (15) is nonsingular.*

Proof:
From Lemma 5.1 we conclude that there exist nonsingular matrices V_d, $d = 1, 2$, such that

$$A = (V_2 \otimes V_1)\Lambda(V_2 \otimes V_1)^{-1},$$

where

$$\Lambda = 4I_{n\times m} + \tilde{\kappa}_1 I_m \otimes \Lambda_1 + \tilde{\kappa}_2 \Lambda_2 \otimes I_n = \mathrm{diag}(\lambda_{11}, \ldots, \lambda_{nm}),$$

$$\Lambda_1 = \mathrm{diag}(\lambda_{1,1}, \ldots, \lambda_{1,n}),$$

and

$$\Lambda_2 = \mathrm{diag}(\lambda_{2,1}, \ldots, \lambda_{2,m}).$$

Now from Lemma 5.1 we get

$$\Re e(\lambda_{kj}) = 4 + \kappa_1 \Re e(\lambda_{1,k}) + \kappa_2 \Re e(\lambda_{2,j}) \geq 4$$

which proves the theorem. $\square$

6 Preconditioners

Holmgren and Otto have investigated semi- and block-circulant preconditioners for Equation (14) showing very good results, [14], [15]. The idea is to create blocks in the preconditioner that are circulant, and then use Fourier-technique to compute the inversion of it. A somewhat similar approach is presented in [11]. There the blocks are Toeplitz with a certain symmetry. By a modified sine transform defined in [10], the solver for the preconditioner system again is based on Fourier-technique. In this report we will employ preconditioners based on Toeplitz blocks. Circulant preconditioners have also been implemented, but since the results from these were inferior to the ones presented here, we do not further discuss such preconditioners in this report. However, we cannot conclude that circulant preconditioners do not work in a domain decomposition context. Numerical spectra indicate that for larger problems they might do, but since we are interested in all problem-sizes, these results are left out.

6.1 Definition of matrices and operators

We start by defining some important matrices , that we will make use of in the following.

Definition 6.1 *In Table 1 we define a number of matrices. A* $\bullet$ *means that the numerical boundary conditions defined in (13) are employed.*

	Boundary-conditions at			
Matrix	$x_1 = 0$	$x_1 = 1$	$x_2 = 0$	$x_2 = 1$
$A_{00}^{(q)}(\bullet)$	*Dirichlet*	$\bullet$	-	-
$A_{00}^{(q)}(D)$	*Dirichlet*	*Dirichlet*	-	-
$A_{11}^{(q)}(\bullet,\bullet)$	*Dirichlet*	$\bullet$	*Dirichlet*	$\bullet$
$A_{11}^{(q)}(D,\bullet)$	*Dirichlet*	*Dirichlet*	*Dirichlet*	$\bullet$
$A_{11}^{(q)}(\bullet,D)$	*Dirichlet*	$\bullet$	*Dirichlet*	*Dirichlet*
$A_{11}^{(q)}(D,D)$	*Dirichlet*	*Dirichlet*	*Dirichlet*	*Dirichlet*

Table 1: Definition of matrices.

□

Next we define the matrix that will be the basis of the preconditioning matrices.

Definition 6.2 R_n *is the matrix of order* n *defined by*

$$R_n = \begin{pmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ -1 & 0 & 1 & 0 & \cdots & 0 \\ \ddots & \ddots & \ddots & \ddots & \cdots & \ddots \\ 0 & \cdots & 0 & -1 & 0 & 1 \\ 0 & \cdots & 0 & 0 & -1 & 0 \end{pmatrix}.$$

□

R_n has the following eigenvalue-decomposition

$$R_n = S_n \Lambda_n^R S_n^H ,$$

where S_n is the modified sine-matrix defined in [10], with entries

$$S_n(k,\ell) = \sqrt{\tfrac{2}{n+1}}\, i^{k+\ell+1} \sin\left(\tfrac{k\ell\pi}{n+1}\right) \quad , \quad k,\ell = 1,\ldots,n \ . \tag{16}$$

The eigenvalue-matrix $\Lambda_n^R = \mathrm{diag}(\lambda_1^R,\ldots,\lambda_n^R)$ is defined by

$$\lambda_k^R = 2i\cos\left(\tfrac{k\pi}{n+1}\right) \quad , \quad k = 1,\ldots,n \ .$$

In the following we will define preconditioners that have a decomposition in modified sine-matrices. We terminate this section by defining some operators that we will make use of.

Definition 6.3 *In Table 2 we define a number of operators to be used.*

Operator	*Operand Q*		*Definition*
	number of blocks	*size of blocks*	
c_{S_n}	1×1	$n \times n$	*Equation (17)*
$c_{I_{m_q}\otimes S_n}$	$m_q \times m_q$	$n \times n$	*Equation (18)*
$c_{S_{m_q}\otimes I_n}$	$m_q \times m_q$	$n \times n$	*Equation (19)*
$c_{S_{m_q}\otimes S_n}$	$m_q \times m_q$	$n \times n$	$c_{S_{m_q}\otimes I_n}(c_{I_{m_q}\otimes S_n}(Q))$

Table 2: Definition of operators.

$$c_{S_n}(Q) = \tfrac{1}{n}\left(\textstyle\sum_{k=1}^{n} q_{k,k}\right) I_n + \tfrac{1}{2(n-1)}\left(\textstyle\sum_{k=1}^{n-1}(q_{k,k+1} - q_{k+1,k})\right) R_n \tag{17}$$

$$c_{I_{m_q}\otimes S_n}(Q) = \begin{pmatrix} c_{S_n}([Q]_{11}) & c_{S_n}([Q]_{12}) & & \\ c_{S_n}([Q]_{21}) & \ddots & \ddots & \\ & \ddots & \ddots & c_{S_n}([Q]_{m_q-1,m_q}) \\ & & c_{S_n}([Q]_{m_q,m_q-1}) & c_{S_n}([Q]_{m_q,m_q}) \end{pmatrix} \tag{18}$$

$$\begin{aligned} c_{S_{m_q}\otimes I_n}(Q) &= \tfrac{1}{m_q}\left[I_n \otimes \left(\textstyle\sum_{j=1}^{m_1}[Q]_{j,j}\right)\right] + \\ &+ \tfrac{1}{2(m_q-1)}\left[R_n \otimes \left(\textstyle\sum_{j=1}^{m_q-1}([Q]_{j,j+1} - [Q]_{j+1,j})\right)\right] \end{aligned} \tag{19}$$

□

6.2 Definition of preconditioners

In this section we define the preconditioners $\hat{C}$, $\hat{A}_{11}$, and $\hat{\mathcal{A}}$. We start with $\hat{A}_{11}$ which is defined by

$$\hat{A}_{11}(\star_1, \star_2) = \mathrm{diag}(\hat{A}^{(1)}(\star_1, \star_2), \ldots, \hat{A}^{(p)}(\star_1, \star_2)), \tag{20}$$

where

$$\left.\begin{array}{l} \hat{A}_{11}^{(q)}(D, \bullet) = c_{I_{m_q} \otimes S_n}(A_{11}^{(q)}(D, \bullet)) \\ \hat{A}_{11}^{(q)}(\bullet, D) = c_{S_{m_q} \otimes I_n}(A_{11}^{(q)}(\bullet, D)) \\ \hat{A}_{11}^{(q)}(D, D) = c_{S_{m_q} \otimes S_n}(A_{11}^{(q)}(D, D)) \end{array}\right. \quad , \quad q = 1, \ldots, p \ ,$$

and $\star_d$, $d = 1, 2$, could be either D or $\bullet$.

The remaining preconditioners are defined by

$$\hat{C}(D) = \hat{A}_{00}(D) - A_{01} \hat{A}_{11}^{-1}(D, D) A_{10},$$

and

$$\hat{\mathcal{A}}(\star_1, \star_2) = \begin{pmatrix} \hat{A}_{00}(D) + A_{01}(D)(\hat{A}_{11}^{-1}(\star_1, \star_2) - \hat{A}_{11}^{-1}(D, D)) A_{10}(D) & A_{01}(D) \\ A_{10}(D) & \hat{A}_{11}(\star_1, \star_2) \end{pmatrix}$$

respectively, where

$$\hat{A}_{00}(D) = c_{I_{p-1} \otimes S_n}(A_{00}(D)).$$

6.3 Preconditioner solve

6.3.1 Schur-Complement preconditioner

From the definition it is clear that $\hat{C}(D)$ has the following decomposition

$$\hat{C}(D) = (I_{p-1} \otimes S_n) U(D) (I_{p-1} \otimes S_n^H). \tag{21}$$

The matrix $U(D)$ is sparse with only three nonzero diagonals

$$U(D) = \begin{pmatrix} \Lambda^{(1)}(D) & \Delta^{(1)}(D) & & \\ \Theta^{(2)}(D) & \ddots & \ddots & \\ & \ddots & \ddots & \Delta^{(p-2)}(D) \\ & & \Theta^{(p-1)}(D) & \Lambda^{(p-1)}(D) \end{pmatrix}, \tag{22}$$

where

$$\left.\begin{array}{rcl} \Lambda^{(q)}(D) & = & \mathrm{diag}(\lambda_1^{(q)}(D), \ldots, \lambda_n^{(q)}(D)) \\ \Delta^{(q)}(D) & = & \mathrm{diag}(\delta_1^{(q)}(D), \ldots, \delta_n^{(q)}(D)) \\ \Theta^{(q)}(D) & = & \mathrm{diag}(\theta_1^{(q)}(D), \ldots, \theta_n^{(q)}(D)) \end{array}\right\} \quad , \quad q = 1, \ldots, p-1 \ .$$

Here

$$\left\{\begin{array}{lcl}
\lambda_k^{(q)}(D) & = & 4+2i\bar{\kappa}_1(D)\cos\left(\frac{k\pi}{n+1}\right)+ \\
 & + & \tilde{\kappa}_{2,\bar{j}}\tilde{\kappa}_{2,\bar{j}+1}\sum_{j=1}^{m_q}\frac{S_{m_q}(m_q,j)S_{m_q}^H(j,m_q)}{4+2i\bar{\kappa}_1(D)\cos\left(\frac{k\pi}{n+1}\right)+2i\bar{\kappa}_2(D,q)\cos\left(\frac{j\pi}{m_q+1}\right)}+ \\
 & + & \tilde{\kappa}_{2,\bar{j}+1}\tilde{\kappa}_{2,\bar{j}+2}\sum_{j=1}^{m_{q+1}}\frac{S_{m_{q+1}}(1,j)S_{m_{q+1}}^H(j,1)}{4+2i\bar{\kappa}_1(D)\cos\left(\frac{k\pi}{n+1}\right)+2i\bar{\kappa}_2(D,q+1)\cos\left(\frac{j\pi}{m_{q+1}+1}\right)} \\
\delta_k^{(q)}(D) & = & -\tilde{\kappa}_{2,\bar{j}+1}\tilde{\kappa}_{2,\bar{j}+m_{q+1}+1}\sum_{j=1}^{m_{q+1}}\frac{S_{m_{q+1}}(1,j)S_{m_{q+1}}^H(j,m_{q+1})}{4+2i\bar{\kappa}_1(D)\cos\left(\frac{k\pi}{n+1}\right)+2i\bar{\kappa}_2(D,q+1)\cos\left(\frac{j\pi}{m_{q+1}+1}\right)} \\
\theta_k^{(q)}(D) & = & -\tilde{\kappa}_{2,\bar{j}+1}\tilde{\kappa}_{2,\bar{j}-m_q+1}\sum_{j=1}^{m_q}\frac{S_{m_q}(m_q,j)S_{m_q}^H(j,1)}{4+2i\bar{\kappa}_1(D)\cos\left(\frac{k\pi}{n+1}\right)+2i\bar{\kappa}_2(D,q)\cos\left(\frac{j\pi}{m_q+1}\right)}
\end{array}\right.$$

$$k=1,\ldots,n \quad , \quad q=1,\ldots,p,$$

where $\bar{j}=q-1+\sum_{\ell=1}^{q}m_\ell$ and

$$\bar{\kappa}_1(D) = \frac{1}{2(n-1)}\sum_{k=1}^{n-1}(\tilde{\kappa}_{1,k}+\tilde{\kappa}_{1,k+1})$$

$$\bar{\kappa}_2(D,q) = \frac{1}{2(m_q-1)}\sum_{j=j_{start}}^{j_{stop}}(\tilde{\kappa}_{2,j}+\tilde{\kappa}_{2,j+1}) \quad , \quad q=1,\ldots,p,$$

$j_{start}=q+\sum_{\ell=1}^{q-1}m_\ell$, and $j_{stop}=q-1+\sum_{\ell=1}^{q}m_\ell$.

From Equation (21) it is clear that the solution of

$$\hat{C}(D)x=y$$

is defined by

Algorithm Schur-Complement preconditioner solve

1. $w=(I_{p-1}\otimes S_n^H)y$
2. Solve $U(D)z=w$
3. $x=(I_{p-1}\otimes S_n)z$

Step 1. and 3. are $p-1$ modified sine transforms and inverse modified sine transforms respectively. The transforms can be computed completely in parallel. In [10] it is shown that each such transform can be accomplished using

$\mathcal{O}(2.5n \log_2 n)$ operations, provided that $n+1$ is a power of 2. The system defined in step 2. and Equation (22), decouples into n systems of order $p-1$. These systems that are spread over the processors, are solved using cyclic reduction, [13]. By symmetry only half of the number of systems have to be solved for, [10].

6.3.2 Subdomain preconditioner

The subdomain preconditioner $\hat{A}_{11}$ is defined in Equation (20). The inversion of $\hat{A}_{11}$ decouples into p independent local solves. Hence they can be performed in parallel with no communication. In [11] it is shown that

$$\hat{A}_{11}^{(q)}(D, \star_2) = (I_{m_q} \otimes S_n) T^{(q)}(D, \star_2)(I_{m_q} \otimes S_n^H),$$

where $\star_2$ is either D or $\bullet$. $T^{(q)}(D, \star_2)$ is sparse with only three nonzero diagonals. Similarly

$$\hat{A}_{11}^{(q)}(\bullet, D) = (S_{m_q} \otimes I_n) T^{(q)}(\bullet, D)(S_{m_q}^H \otimes I_n),$$

where $T^{(q)}(\bullet, D)$ has a similar sparsity pattern as $T^{(q)}(D, \star_2)$. Hence

$$\hat{A}_{11}^{(q)} x_1^{(q)} = y_1^{(q)} \tag{23}$$

can be solved as

Algorithm Subdomain preconditioner solve a)

1. $w = (I_{m_q} \otimes S_n^H) y_1^{(q)}$
2. Solve $T^{(q)}(D, \star_2) z = w$
3. $x_1^{(q)} = (I_{m_q} \otimes S_n) z$

Algorithm Subdomain preconditioner solve b)

1. $w = (S_{m_q}^H \otimes I_n) y_1^{(q)}$
2. Solve $T^{(q)}(\bullet, D) z = w$
3. $x_1^{(q)} = (S_{m_q} \otimes I_n) z$

By the previous section it is clear that step 1. and 3. are m_q/n transforms requiring $\mathcal{O}(2.5n \log_2 n)/\mathcal{O}(2.5 m_q \log_2 m_q)$ operations each. Step 2. yields the solution of n/m_q tridiagonal systems of order m_q/n. By symmetry we only have to solve for half of the number of systems.

6.3.3 Partitioned Matrix preconditioner

The solution of

$$\hat{\mathcal{A}}(\star_1, \star_2) \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \end{pmatrix}$$

is defined by block Gaussian elimination yielding

Algorithm Partitioned Matrix preconditioner solve

1. Solve $\hat{A}_{11}(\star_1, \star_2)w = y_1$
2. Solve $\hat{C}(D)x_0 = y_0 - A_{01}w$
3. Solve $\hat{A}_{11}(\star_1, \star_2)v = -A_{10}x_0$
4. $x_1 = w + v$.

Hence the inversion of $\hat{\mathcal{A}}$ is defined by **Algorithm Partitioned Matrix preconditioner solve** and Section 6.3.1 and 6.3.2.

7 Theoretical analysis

In this section we will theoretically analyze the spectra of both the preconditioned system and the original system for some special cases. We here consider Equation (12) with $\sigma_1 = \sigma_2 = 1$. Initially, we state a lemma that we will make use of in the following. The proof can be found in [12].

Lemma 7.1 *Let α be a complex number with $\Re e\ \alpha > 0$, $\beta > 0$ real, and $N + 1 = 2^k$. Then*

$$\begin{aligned} \Phi_{k\ell}(\alpha, \beta, N) &\equiv \frac{2}{N+1} \sum_{j=1}^{N} \frac{\sin\left(\frac{kj\pi}{N+1}\right) \sin\left(\frac{\ell j\pi}{N+1}\right)}{\alpha + 2\beta \cos\left(\frac{j\pi}{N+1}\right)} = \\ &= \frac{i}{\beta} \left(\frac{z_\Phi^{k+\ell} - z_\Phi^{|k-\ell|}}{z_\Phi - z_\Phi^{-1}} + \frac{(z_\Phi^{-k} - z_\Phi^{k})(z_\Phi^{-\ell} - z_\Phi^{\ell})}{z_\Phi - z_\Phi^{-1}} \cdot \frac{z_\Phi^{2N+2}}{1 - z_\Phi^{2N+2}} \right), \end{aligned} \tag{24}$$

where z_Φ is the root with $|z_\Phi| < 1$ to

$$z^2 + \frac{\alpha}{\beta} z + 1 = 0.$$

$\square$

7.1 SCM

7.1.1 The Schur-Complement system for a semi-periodic problem

Here we theoretically analyze the semi-periodic problem, i.e., we impose periodic boundary conditions in one space-direction. We replace (13) by

$$\begin{cases} \dfrac{\partial u}{\partial x_1} \approx D_{0,x_1} u_{k,j} & \\ & , \quad k = 1, \dots, n-1 \text{ and } j = 1, \dots, m-1 \\ \dfrac{\partial u}{\partial x_2} \approx D_{0,x_2} u_{k,j} & \\ \dfrac{\partial u}{\partial x_1} \approx \dfrac{u_{1,j} - u_{n-1,j}}{2h_1} & , \quad j = 1, \dots, m \\ \dfrac{\partial u}{\partial x_2} \approx D_{-,x_2} u_{k,m} & , \quad k = 1, \dots, n. \end{cases}$$

We consider the two-domain decomposition, and restrict our formulation to the case $m_1 = m_2$. The latter condition is not necessary, it is just to simplify the notation. The coefficient-matrices then become

$$A_{00} = \tilde{D}, \tag{25}$$

$$A_{11}^{(1)} = \begin{pmatrix} \tilde{D} & G & & & \\ -G & \tilde{D} & G & & \\ & \ddots & \ddots & \ddots & \\ & & -G & \tilde{D} & G \\ & & & -G & \tilde{D} \end{pmatrix}, \tag{26}$$

and

$$A_{11}^{(2)} = \begin{pmatrix} \tilde{D} & G & & & \\ -G & \tilde{D} & G & & \\ & \ddots & \ddots & \ddots & \\ & & -G & \tilde{D} & G \\ & & & -2G & \tilde{D} + 2G \end{pmatrix}, \tag{27}$$

where

$$\tilde{D} = \begin{pmatrix} 4 & \kappa_1 & & -\kappa_1 \\ -\kappa_1 & \ddots & \ddots & \\ & \ddots & \ddots & \kappa_1 \\ \kappa_1 & & -\kappa_1 & 4 \end{pmatrix},$$

and $G = \kappa_2 I_n$. We also have

$$A_{01}^{(11)} = \begin{pmatrix} 0 & \cdots & 0 & -G \end{pmatrix},$$

$$A_{01}^{(12)} = \begin{pmatrix} G & 0 & \cdots & 0 \end{pmatrix},$$

$$A_{10}^{(11)} = -(A_{01}^{(11)})^T,$$

and

$$A_{10}^{(21)} = -(A_{10}^{(21)})^T.$$

We are going to study the eigenvalue-decomposition of C defined in (5), when A_{11} is defined through (26) and (27). We start by considering the eigenvalue-decomposition of $A_{11}^{(q)}$, $q = 1, 2$.

$A_{11}^{(1)}$ defined in (26) is block-Toeplitz with circulant blocks, which means that it has an eigenvalue-decomposition as

$$A_{11}^{(1)} = (S_{m_1} \otimes F_n)\Lambda^{(1)}(S_{m_1}^H \otimes F_n^H),$$

where F_n is the Fourier-matrix defined by

$$F_n(k, \ell) = \sqrt{\tfrac{1}{n}} e^{\left(\frac{2\pi i (k-1)(\ell-1)}{n}\right)} \quad , \quad k, \ell = 1, \ldots, n \ .$$

The eigenvalue-matrix $\Lambda^{(1)}$ is defined by

$$\Lambda^{(1)} = \mathrm{diag}(\Lambda_1^{(1)}, \ldots, \Lambda_{m_1}^{(1)}),$$

where

$$\Lambda_j^{(1)} = \mathrm{diag}(\lambda_{1j}^{(1)}, \ldots, \lambda_{nj}^{(1)}),$$

and

$$\lambda_{kj}^{(1)} = 4 + 2i\kappa_1 \sin\left(\tfrac{2(k-1)\pi}{n}\right) + 2i\kappa_2 \cos\left(\tfrac{j\pi}{m_1+1}\right) \quad , \quad k = 1, \ldots, n, j = 1, \ldots, m_1 \ .$$

Since it is the eigenvalue-decomposition of C we really are interested in, we observe that the term $-A_{01}^{(1)}(A_{11}^{(1)})^{-1}A_{10}^{(1)}$ can be written as

$$\begin{aligned} & -\kappa_2^2(u \otimes I_n)(S_{m_1} \otimes F_n)(\Lambda^{(1)})^{-1}(S_{m_1}^H \otimes F_n^H)(u \otimes I_n) = \\ = \quad & F_n(-\kappa_2^2 \sum_{j=1}^{m_1} S_{m_1}(m_1, j) S_{m_1}^H(j, m_1)(\Lambda_j^{(1)})^{-1}) F_n^H \equiv F_n E^{(1)} F_n^H, \end{aligned}$$

where

$$u = \begin{pmatrix} 0 & \dots & 0 & 1 \end{pmatrix}^T. \tag{28}$$

$E^{(1)} = \mathrm{diag}(e_1^{(1)}, \dots, e_n^{(1)})$, where

$$e_k^{(1)} = -\kappa_2^2 \sum_{j=1}^{m_1} \frac{S_{m_1}(m_1, j) S_{m_1}^H(j, m_1)}{4 + 2i\kappa_1 \sin\left(\frac{2(k-1)\pi}{n}\right) + 2i\kappa_2 \cos\left(\frac{j\pi}{m_1+1}\right)} \quad , \quad k = 1, \dots, n.$$

By Lemma 7.1 we get

$$\begin{array}{rcl} e_k^{(1)} & = & -\kappa_2 \Phi_{m_1 m_1}(4 + 2i\kappa_1 \sin\left(\frac{2(k-1)\pi}{n}\right), i\kappa_2, m_1) = \\ & = & -\kappa_2 i z_k \dfrac{1 - z_k^{2m_1}}{1 - z_k^{2m_1+2}} \end{array} \quad , \quad k = 1, \dots, n$$

$$\begin{array}{l} z_k = z_k^a \quad , \quad 2 \le k \le \frac{n+2}{2} \\ z_k = z_k^b \quad , \quad k = 1 \text{ and } \frac{n+2}{2} < k \le n, \end{array} \tag{29}$$

where

$$\begin{array}{rcl} z_k^a & = & \frac{2i}{\kappa_2} - \frac{\kappa_1}{\kappa_2} \sin\left(\frac{2(k-1)\pi}{n}\right) + \sqrt{\left(\frac{2i}{\kappa_2} - \frac{\kappa_1}{\kappa_2} \sin\left(\frac{2(k-1)\pi}{n}\right)\right)^2 - 1} \\ z_k^b & = & \frac{2i}{\kappa_2} - \frac{\kappa_1}{\kappa_2} \sin\left(\frac{2(k-1)\pi}{n}\right) - \sqrt{\left(\frac{2i}{\kappa_2} - \frac{\kappa_1}{\kappa_2} \sin\left(\frac{2(k-1)\pi}{n}\right)\right)^2 - 1}. \end{array} \tag{30}$$

Following [17] it is clear that $|z_k| < 1$. As $m_2 = m_1$ we express $A_{11}^{(2)}$ as $A_{11}^{(2)} = A_{11}^{(1)} + UV^T$, where

$$\begin{array}{rcl} U & = & \kappa_2(u \otimes I_n), \\ V & = & v \otimes I_n, \\ v & = & \begin{pmatrix} 0 & \dots & 0 & -1 & 2 \end{pmatrix}^T, \end{array} \tag{31}$$

and u is defined in (28). Using the Sherman-Morrison-Woodbury formula we obtain

$$(A_{11}^{(1)} + UV^T)^{-1} = (A_{11}^{(1)})^{-1} - (A_{11}^{(1)})^{-1} U (I_n + V^T (A_{11}^{(1)})^{-1} U)^{-1} V^T (A_{11}^{(1)})^{-1}. \tag{32}$$

Now

$$
\begin{aligned}
& I_n + V^T (A_{11}^{(1)})^{-1} U = \\
= \;& I_n + \kappa_2 (v^T \otimes I_n)(S_{m_1} \otimes F_n)(\Lambda^{(1)})^{-1}(S_{m_1}^H \otimes F_n^H)(u \otimes I_n) = \\
= \;& I_n + \kappa_2 (v^T S_{m_1} \otimes F_n)(\Lambda^{(1)})^{-1}(S_{m_1}^H u \otimes F_n^H) = \\
= \;& F_n (I_n + \kappa_2 \sum_{j=1}^{m_1} (-S_{m_1}(m_1-1,j) + 2S_{m_1}(m_1,j)) S_{m_1}^H (j,m_1)(\Lambda_j^{(1)})^{-1}) F_n^H \equiv \\
\equiv \;& F_n L F_n^H .
\end{aligned}
\tag{33}
$$

Here $L = \mathrm{diag}(l_1, \ldots, l_n)$ and

$$
l_k = 1 + \kappa_2 \sum_{j=1}^{m_1} \frac{(-S_{m_1}(m_1-1,j) + 2S_{m_1}(m_1,j)) S_{m_1}^H (j,m_1)}{4 + 2i\kappa_1 \sin\left(\frac{2(k-1)\pi}{n}\right) + 2i\kappa_2 \cos\left(\frac{j\pi}{m_1+1}\right)} \quad , \quad k = 1, \ldots, n.
\tag{34}
$$

Using Lemma 7.1, Equation (34) yields

$$
\begin{aligned}
l_k \;&=\; 1 + \kappa_2 \times \\
&\times\; \left(i\Phi_{m_1-1,m_1}(4 + 2i\kappa_1 \sin\left(\tfrac{2(k-1)\pi}{n}\right), i\kappa_2, m_1) + \right. \\
&+\; \left. 2\Phi_{m_1 m_1}(4 + 2i\kappa_1 \sin\left(\tfrac{2(k-1)\pi}{n}\right), i\kappa_2, m_1) \right) = \qquad , \quad k = 1, \ldots, n. \\
&=\; -z_k^2 \frac{1 - z_k^{2m_1-2}}{1 - z_k^{2m_1+2}} + 2iz_k \frac{1 - z_k^{2m_1}}{1 - z_k^{2m_1+2}}
\end{aligned}
\tag{35}
$$

where z_k is defined in (29) and (30). Inserting (33) in (32) yields

$$
(A_{11}^{(1)} + UV^T)^{-1} = (S_{m_1} \otimes F_n)((\Lambda^{(1)})^{-1} - \kappa_2 (\Lambda^{(1)})^{-1} W (\Lambda^{(1)})^{-1})(S_{m_1}^H \otimes F_n^H),
$$

where W is defined by

$$
\begin{aligned}
W \;&=\; (S_{m_1}^H \otimes F_n^H)(u \otimes I_n) F_n L^{-1} F_n^H (v \otimes I_n)(S_{m_1} \otimes F_n) = \\
&=\; S_{m_1}^H u v^T S_{m_1} \otimes L^{-1}.
\end{aligned}
\tag{36}
$$

Now $-A_{01}^{(2)}(A_{11}^{(2)})^{-1} A_{10}^{(2)}$ can be written as

$$
-\kappa_2^2 (w^T \otimes I_n)(A_{11}^{(1)} + UV^T)^{-1}(w \otimes I_n),
$$

where

$$
w = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \end{pmatrix}^T ,
$$

which together with Equation (36) yields

$$-A_{01}^{(2)}(A_{11}^{(2)})^{-1}A_{10}^{(2)} = F_n E^{(2)} F_n^H,$$

where $E^{(2)}$ is defined by

$$\begin{aligned}
E^{(2)} &= -\kappa_2^2 \Big(\textstyle\sum_{j=1}^{m_1} S_{m_1}(1,j)S_{m_1}^H(j,1)(\Lambda_j^{(1)})^{-1} - \\
&- \kappa_2 L^{-1} \textstyle\sum_{j=1}^{m_1} S_{m_1}(1,j)S_{m_1}^H(j,m_1)(\Lambda_j^{(1)})^{-1}\times \\
&\times \textstyle\sum_{j=1}^{m_1} S_{m_1}^H(j,1)(-S_{m_1}(m_1-1,j)+2S_{m_1}(m_1,j))(\Lambda_j^{(1)})^{-1}\Big) \equiv \\
&\equiv \operatorname{diag}(e_1^{(2)},\ldots,e_n^{(2)}).
\end{aligned}$$

The sums in $e_k^{(2)}$, $k=1,\ldots,n$, can be computed using Lemma 7.1, and by inserting (35) we get

$$\begin{aligned}
e_k^{(2)} &= \kappa_2\left(\frac{iz_k(1-z_k^{2m_1})}{1-z_k^{2m_1+2}}+\right. \\
&+ \left.\frac{iz_k^{2m_1-1}(1-z_k^2)(1-z_k^4)+2z_k^{2m_1}(1-z_k^2)^2}{(1-z_k^{2m_1+2})(1-z_k^{2m_1+2}-z_k^2(1-z_k^{2m_1-2})+2iz_k(1-z_k^{2m_1}))}\right)
\end{aligned}$$

$$k=1,\ldots,n.$$

Finally A_{00} defined in (25), has a well-known decomposition in Fourier-matrices, and hence the eigenvalues of the semi-periodic problem divided in 2 subdomains gives a Schur-Complement C with an eigenvalue-decomposition

$$C = F_n \Lambda F_n^H, \tag{37}$$

with eigenvalues

$$\begin{aligned}
\lambda_k &= 4 + 2i\kappa_1 \sin\left(\tfrac{2(k-1)\pi}{n}\right) + \\
&+ \kappa_2\left(2iz_k\frac{1-z_k^{2m_1}}{1-z_k^{2m_1+2}}+\right. \\
&+ \left.\frac{iz_k^{2m_1-1}(1-z_k^2)(1-z_k^4)+2z_k^{2m_1}(1-z_k^2)^2}{(1-z_k^{2m_1+2})(1-z_k^{2m_1+2}-z_k^2(1-z_k^{2m_1-2})+2iz_k(1-z_k^{2m_1}))}\right),
\end{aligned} \tag{38}$$

where z_k is defined in (29) and (30).

From Equation (37) we see that the solution of the Schur-Complement system for a semi-periodic problem can be obtained from one Fourier transform, the solution of a diagonal system of order n, and finally one inverse Fourier transform. This can be generalized to an arbitrary number of subdomains. Then we perform $p-1$ Fourier transforms in parallel, solve n tridiagonal systems of order $p-1$, and finally compute $p-1$ parallel inverse Fourier transforms. The solution of the tridiagonal systems can be parallelized using for example cyclic reduction.

From Equation (38) we will derive a theorem concerning the asymptotic spectrum. Note that we do not make any assumptions about the relation between the time-step and the space-step, which is the case in [17] and [12].

Theorem 7.2 *Assume that*

$$\phi \equiv \frac{n}{m} < 1, \tag{39}$$

and that κ_1 is large and fixed. In the limit $n \to \infty$ the eigenvalues λ_k, $k = 1, \ldots, n$, defined in (38) reside on a curve-segment $\tilde{\lambda}(\zeta)$ defined by

$$\tilde{\lambda}(\zeta) = 2\kappa_1 \phi^{-1}\sqrt{1-\zeta^2} + \frac{4i\zeta}{\sqrt{1-\zeta^2}} + \mathcal{O}(\kappa_1^{-1}) \quad , \quad -\phi \le \zeta \le \phi \ . \tag{40}$$

Proof:
Let δ_k and γ_k be defined by

$$\begin{aligned} \delta_k &= \phi \sin\left(\frac{2(k-1)\pi}{n}\right) \\ \gamma_k &= \frac{2i\phi}{\kappa_1} - \delta_k . \end{aligned}$$

Then z_k^a and z_k^b defined in (30) can be expressed as

$$\begin{aligned} z_k^a &= \gamma_k + \sqrt{\gamma_k^2 - 1} \\ z_k^b &= \gamma_k - \sqrt{\gamma_k^2 - 1}. \end{aligned}$$

For large κ_1 we get from a Taylor expansion that

$$z_k = -\delta_k - i\sqrt{1-\delta_k^2} + \frac{2i\phi}{\kappa_1} + \frac{2\delta_k\phi}{\kappa_1\sqrt{1-\delta_k^2}} + \mathcal{O}(\kappa_1^{-2}) \quad , \quad k = 1, \ldots, n.$$

Since

$$|z_k|^2 = 1 - \frac{4\phi}{\kappa_1\sqrt{1-\delta_k^2}} + \mathcal{O}(\kappa_1^{-2}) < 1$$

we get

$$\lim_{n\to\infty} |z_k|^{2m_1} = 0,$$

and hence, from Equation (38) we get (40). □

From Theorem 7.6 we see that the eigenvalues stay bounded and are well separated from the origin when the problem-size increases. In Figures 2 and 3 we show how well the asymptotic formula agrees with the eigenvalues for large problems.

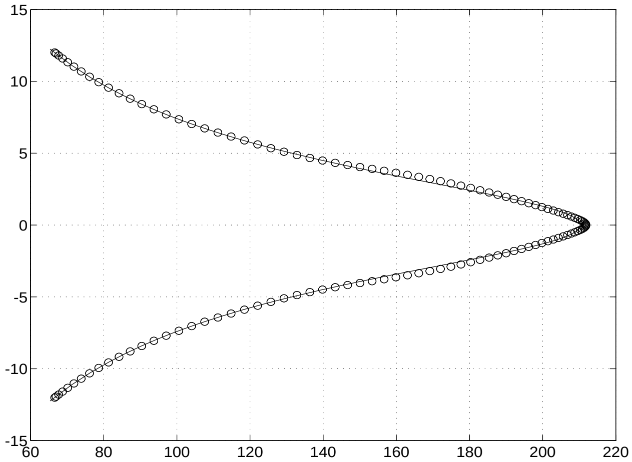

Figure 2: Asymptotic spectrum solid line and eigenvalues marked with circles for $\kappa_1 = 100$, $n = 250$ and $m = 263$, i.e. $\phi = 0.95$.

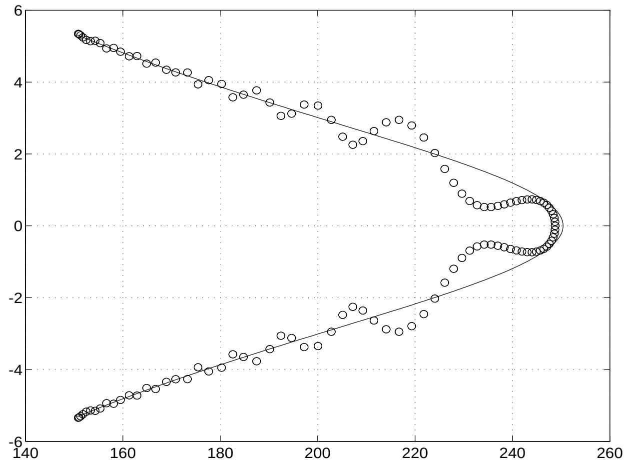

Figure 3: Asymptotic spectrum solid line and eigenvalues marked with circles for $\kappa_1 = 100$, $n = 250$ and $m = 313$, i.e. $\phi = 0.80$.

We now enclose the asymptotic spectrum $\tilde{\lambda}$, defined in (40), in a semi-circle $\mathcal{C}^-(c,R)$. Here $\mathcal{C}(c,R)$ denotes the circle with center c and radius R, and $\mathcal{C}^-(c,R)$ is the semi-circle defined by

$$\{z|(z \in \mathcal{C}(c,R) \text{ and } \Re e z \le c) \text{ or } (\Re e z = c \text{ and } |\Im m z| \le R)\}.$$

In the following we neglect $\mathcal{O}(\kappa_1^{-1})$-terms.

Lemma 7.3 *The semi-circle $\mathcal{C}^-(c,R)$, where c and R are defined by*

$$\begin{aligned} c &= \frac{2\kappa_1}{\phi}, \\ R &= |\tilde{\lambda}(\pm\phi) - c| = \sqrt{\frac{16\phi^2}{1-\phi^2} + \frac{4\kappa_1^2(2-\phi^2-2\sqrt{1-\phi^2})}{\phi^2}} \end{aligned}$$

encloses the asymptotic spectrum $\tilde{\lambda}$ defined in (40).

Proof:
Define $r(\zeta)$ as the distance between c and $\tilde{\lambda}(\zeta)$. Then

$$\begin{aligned} r^2 &= \frac{16\zeta^2}{1-\zeta^2} + \frac{4\kappa_1^2(1-\sqrt{1-\zeta^2})^2}{\phi^2} \\ &= \frac{16\zeta^2}{1-\zeta^2} + \frac{4\kappa_1^2(2-\zeta^2-2\sqrt{1-\zeta^2})}{\phi^2} \end{aligned}$$

and

$$\frac{d^2r^2}{d\zeta^2} = \frac{32(1-\zeta^2)^2 - 128(\zeta^4-\zeta^2)}{(1-\zeta^2)^4} + \frac{4\kappa_1^2}{\phi^2}\left(-2 + \frac{2(1+\zeta^2-\zeta^4)}{\sqrt{1-\zeta^2}}\right) > 0.$$

Hence, r^2 obtains its maximum value at the endpoints yielding $r(\zeta) \le R$. Finally

$$\Re e\ \tilde{\lambda}(\zeta) = \frac{2\kappa_1\sqrt{1-\zeta^2}}{\phi} \le \frac{2\kappa_1}{\phi}$$

which proves the theorem. □

From Lemma 7.3 and the general result in [18] we derive the following theorem.

Theorem 7.4 *The asymptotic convergence factor ρ defined in (10), fulfills*

$$\rho \le \sqrt{\frac{4\phi^4}{\kappa_1^2(1-\phi^2)} + 2 - \phi^2 - 2\sqrt{1-\phi^2}}. \tag{41}$$

Proof:
From [18] we get that if the spectrum is enclosed in a circle $\mathcal{C}(c, R)$, then the asymptotic convergence factor fulfills

$$\rho \leq \frac{R}{c}.$$

With this result and Lemma 7.3 we get (41). $\square$

Now define the *residual reduction* $\tilde{\rho}_k$ as

$$\tilde{\rho}_k \equiv \left(\frac{\|r_k\|_2}{\|r_0\|_2} \right)^{1/k}. \tag{42}$$

From (9), (37), (42), and the fact that the Fourier-matrix is unitary, we conclude that

$$\tilde{\rho}_\infty \leq \rho.$$

In Figure 4 we display the asymptotic convergence factor as a function of ϕ, for $\kappa_1 = 100$. In the same figure we present $\tilde{\rho}_{20}$ for the original problem (13).

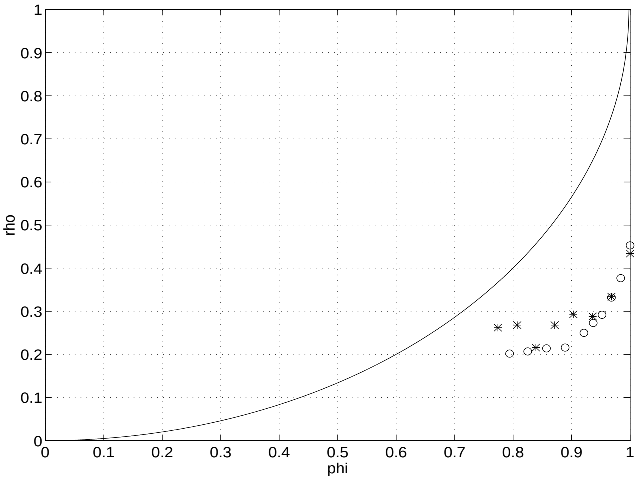

Figure 4: Asymptotic convergence factor solid line. The residual reduction $\tilde{\rho}_{20}$ is marked with stars for $m = 31$, and circles for $m = 63$.

From Figure 4 we conclude that the shape of the obtained residual reduction agrees very well with the asymptotic convergence factor, especially for the larger problem. Hence we can expect an extensive gain in time by simply let the aspect ration of the space-discretization slightly decrease from 1.

7.1.2 The preconditioned Schur-Complement system in 1D

The 2D-problem for an arbitrary number of subdomains is difficult to analyze. Hence we will study the problem in one space-dimension. We consider the line oriented in the x_1-direction for $0 \leq x_1 \leq 1$, that we divide in p line-segments. Then the coefficient-matrices become

$$
\begin{aligned}
A_{11}^{(q)} &= \begin{pmatrix} 4 & \kappa_1 & & \\ -\kappa_1 & \ddots & \kappa_1 & \\ & \ddots & \ddots & \kappa_1 \\ & & -\kappa_1 & 4 \end{pmatrix} & , \quad q = 1, \ldots, p-1 \\
A_{11}^{(p)} &= \begin{pmatrix} 4 & \kappa_1 & & \\ -\kappa_1 & \ddots & \kappa_1 & \\ & \ddots & \ddots & \kappa_1 \\ & & -2\kappa_1 & 4+2\kappa_1 \end{pmatrix} & \\
A_{00}^{(q)} &= 4 & , \quad q = 1, \ldots, p-1 \\
A_{01}^{(qq)} &= \begin{pmatrix} 0 & \cdots & 0 & -\kappa_1 \end{pmatrix} & , \quad q = 1, \ldots, p-1 \\
A_{01}^{(q(q+1))} &= \begin{pmatrix} \kappa_1 & 0 & \cdots & 0 \end{pmatrix} & , \quad q = 1, \ldots, p-1 \\
A_{10}^{(qq)} &= \begin{pmatrix} 0 & \cdots & 0 & \kappa_1 \end{pmatrix}^T & , \quad q = 1, \ldots, p-1 \\
A_{10}^{((q+1)q)} &= \begin{pmatrix} -\kappa_1 & 0 & \cdots & 0 \end{pmatrix}^T & , \quad q = 1, \ldots, p-1.
\end{aligned}
\tag{43}
$$

The matrices defined in (43) yields a Schur-Complement C as

$$C = \hat{C} + C',$$

where

$$\hat{C} = \begin{pmatrix} \hat{\lambda} & \hat{\delta} & & \\ \hat{\theta} & \ddots & \ddots & \\ & \ddots & \ddots & \hat{\delta} \\ & & \hat{\theta} & \hat{\lambda} \end{pmatrix}, \tag{44}$$

$$\hat{\lambda} = 4 + \kappa_1^2 \sum_{k=1}^{\bar{n}} \frac{S_{\bar{n}}(\bar{n},k) S_{\bar{n}}^H(k,\bar{n})}{4 + 2i\kappa_1 \cos\left(\frac{k\pi}{\bar{n}+1}\right)} + \kappa_1^2 \sum_{k=1}^{\bar{n}} \frac{S_{\bar{n}}(1,k) S_{\bar{n}}^H(k,1)}{4 + 2i\kappa_1 \cos\left(\frac{k\pi}{\bar{n}+1}\right)}, \tag{45}$$

$$\hat{\delta} = -\kappa_1^2 \sum_{k=1}^{\bar{n}} \frac{S_{\bar{n}}(1,k)S_{\bar{n}}^H(k,\bar{n})}{4 + 2i\kappa_1 \cos\left(\frac{k\pi}{\bar{n}+1}\right)}, \tag{46}$$

$$\hat{\theta} = -\kappa_1^2 \sum_{k=1}^{\bar{n}} \frac{S_{\bar{n}}(\bar{n},k)S_{\bar{n}}^H(k,1)}{4 + 2i\kappa_1 \cos\left(\frac{k\pi}{\bar{n}+1}\right)}, \tag{47}$$

and

$$C' = \begin{pmatrix} 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & \hat{\lambda}' \end{pmatrix}.$$

Here $\bar{n} = \frac{n+1}{p} - 1$.

We start by showing how to evaluate $\hat{\lambda}$, $\hat{\delta}$, and $\hat{\theta}$.

Theorem 7.5 *Under the assumption $\bar{n}+1 = 2^k$, $k \geq 2$, $\hat{\lambda}$, $\hat{\delta}$ and $\hat{\theta}$ defined in (45), (46) and (47) satisfies*

$$\begin{aligned} \hat{\lambda} &= 4 + 2i\kappa_1 z_{\hat{C}} \frac{1 - z_{\hat{C}}^{2\bar{n}}}{1 - z_{\hat{C}}^{2\bar{n}+2}} \\ \hat{\delta} &= i\kappa_1 z_{\hat{C}}^{\bar{n}} \frac{1 - z_{\hat{C}}^{2}}{1 - z_{\hat{C}}^{2\bar{n}+2}} \qquad , \quad z_{\hat{C}} = i\left(\frac{2}{\kappa_1} - \sqrt{\frac{4}{\kappa_1^2} + 1}\right) . \\ \hat{\theta} &= i\kappa_1 z_{\hat{C}}^{\bar{n}} \frac{1 - z_{\hat{C}}^{2}}{1 - z_{\hat{C}}^{2\bar{n}+2}} \end{aligned} \tag{48}$$

Proof:
From the definition of the modified sine-matrix (16), we get

$$S_{\bar{n}}(\bar{n},k)S_{\bar{n}}^H(k,\bar{n}) = \tfrac{2}{\bar{n}+1} \sin^2\left(\tfrac{\bar{n}k\pi}{\bar{n}+1}\right)$$

$$S_{\bar{n}}(1,k)S_{\bar{n}}^H(k,1) = \tfrac{2}{\bar{n}+1} \sin^2\left(\tfrac{k\pi}{\bar{n}+1}\right).$$

Thus

$$\hat{\lambda} = 4 + \kappa_1^2 \left(\Phi_{11}(4, i\kappa_1, \bar{n}) + \Phi_{\bar{n}\bar{n}}(4, i\kappa_1, \bar{n})\right),$$

where $\Phi_{kl}(\alpha, \beta, \bar{n})$ is defined in Equation (28). According to Lemma 7.1

$$\Phi_{11}(4, i\kappa_1, \bar{n}) = \Phi_{\bar{n}\bar{n}}(4, i\kappa_1, \bar{n}) = \frac{i}{\kappa_1} z_{\hat{C}} \frac{1 - z_{\hat{C}}^{2\bar{n}}}{1 - z_{\hat{C}}^{2\bar{n}+2}},$$

$$z_{\hat{C}} = i\left(\frac{2}{\kappa_1} - \sqrt{\frac{4}{\kappa_1^2}+1}\right),$$

which gives $\hat{\lambda}$. Now

$$S_{\bar{n}}(1,k)S_{\bar{n}}^H(k,\bar{n}) = -\tfrac{2}{\bar{n}+1}\sin\left(\tfrac{\bar{n}k\pi}{\bar{n}+1}\right)\sin\left(\tfrac{k\pi}{\bar{n}+1}\right),$$

and hence

$$\hat{\delta} = -\kappa_1^2\Phi_{1\bar{n}}(4, i\kappa_1, \bar{n}) = i\kappa z_{\hat{C}}^{\bar{n}}\frac{1-z_{\hat{C}}^2}{1-z_{\hat{C}}^{2\bar{n}+2}}.$$

The proof for $\hat{\theta}$ is similar to the one for $\hat{\delta}$ and is left out.

Now it remains to prove that $z_{\hat{C}}$ is a root to $z^2 + \frac{\alpha}{\beta}z + 1 = 0$ and that $\left|z_{\hat{C}}\right| < 1$. As $\alpha = 4$ and $\beta = i\kappa_1$ it is clear that $z_{\hat{C}}$ is a root and since $\frac{2}{\kappa_1} > 0$, we get $\sqrt{\frac{4}{\kappa_1^2}+1} < 1 + \frac{2}{\kappa_1}$, and finally $|z_{\hat{C}}| < 1$ which completes the proof. $\square$

We want to derive an expression for $\hat{\lambda}'$. We have

$$\hat{\lambda} + \hat{\lambda}' = 4 - A_{01}^{(p-2,p-1)}(A_{11}^{(p-1,p-1)})^{-1}A_{10}^{(p-1,p-2)} - A_{01}^{(p-1,p)}(A_{11}^{(pp)})^{-1}A_{10}^{(p,p-1)}.$$

For simplicity we introduce

$$A = A_{11}^{(q)} \quad , \quad q = 1, \ldots, p-1.$$

Employing the Sherman-Morrison formula we get

$$(A_{11}^{(pp)})^{-1} = A^{-1} - \kappa_1\frac{A^{-1}uv^TA^{-1}}{1+v^TA^{-1}u},$$

where u and v are defined in (28) and (31). Hence,

$$\begin{aligned}\hat{\lambda} + \lambda' &= 4 - A_{01}^{(p-2,p-1)}A^{-1}A_{10}^{(p-1,p-2)} - A_{01}^{(p-1,p)}A^{-1}A_{10}^{(p,p-1)} + \\ &+ \kappa_1 A_{01}^{(p-1,p)}\frac{A^{-1}uv^TA^{-1}}{1+v^TA^{-1}u}A_{10}^{(p,p-1)},\end{aligned}$$

which yields

$$\hat{\lambda}' = \kappa_1 A_{01}^{(p-1,p)}\frac{A^{-1}uv^TA^{-1}}{1+v^TA^{-1}u}A_{10}^{(p,p-1)}.$$

Using that $A^{-1} = S_{\bar{n}}\hat{\Lambda}S_{\bar{n}}^H$, where $\hat{\Lambda} = \mathrm{diag}(\hat{\lambda}_1, \ldots, \hat{\lambda}_{\bar{n}})$, and

$$\hat{\lambda}_k = 4 + 2i\kappa_1\cos\left(\tfrac{k\pi}{\bar{n}+1}\right) \quad , \quad k = 1, \ldots, \bar{n} \ ,$$

and $S_{\bar{n}}$ is the modified sine-matrix, we get from straightforward calculations that

$$\hat{\lambda}' = \frac{\kappa_1^3(-2\Phi_{\bar{n}1}(4,\kappa_1,\bar{n}) - i\Phi_{\bar{n}-1,1}(4,\kappa_1,n))\Phi_{1\bar{n}}(4,\kappa_1,\bar{n})}{1+\kappa_1(i\Phi_{\bar{n}-1,\bar{n}}(4,\kappa_1,\bar{n}) + 2\Phi_{\bar{n}\bar{n}}(4,\kappa_1,\bar{n}))} =$$

$$= -\kappa_1 \frac{i z_{\hat{C}}^{2\bar{n}-1}(1-z_{\hat{C}}^2)^2(1+z_{\hat{C}}^2 - 2iz_{\hat{C}})}{(1-z_{\hat{C}}^{2\bar{n}+2})((z_{\hat{C}}-i)^2 + z_{\hat{C}}^{2\bar{n}}(z_{\hat{C}}+i)^2)},$$

where $z_{\hat{C}}$ is defined in (48).

We will derive the eigenvalues of the preconditioned system, when the preconditioner is obtained from Toeplitz approximations. This yields the preconditioner $\hat{C}$, defined in (44). Since $\hat{\theta} = \hat{\delta}$, this matrix is symmetric tridiagonal and we get

$$\hat{C} = \dot{S}_{p-1}\hat{\hat{\Lambda}}\dot{S}_{p-1}^H,$$

where

$$\dot{S}_{p-1}(j,\ell) = \sqrt{\tfrac{2}{p}}\sin\left(\tfrac{k\ell\pi}{p}\right) \quad , \quad j,\ell = 1,\ldots,p-1,$$

and

$$\hat{\hat{\Lambda}} = \mathrm{diag}(\hat{\hat{\lambda}}_1,\ldots,\hat{\hat{\lambda}}_{p-1}),$$

where

$$\hat{\hat{\lambda}}_j = \hat{\lambda} + 2\hat{\delta}\cos\left(\tfrac{j\pi}{p}\right) \quad , \quad j = 1,\ldots,p-1.$$

The coefficient-matrix to the preconditioned system is

$$\hat{C}^{-1}C = I + \hat{C}^{-1}C',$$

and in order to derive the eigenvalues of $\hat{C}^{-1}C$ we consider the part $\hat{C}^{-1}C'$. We get

$$\hat{C}^{-1}C' = \begin{pmatrix} 0 & \cdots & 0 & \hat{\lambda}'\mu_1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & \hat{\lambda}'\mu_{p-1} \end{pmatrix},$$

where

$$\mu_s = \sum_{q=1}^{p-1} \frac{\dot{S}(s,q)\dot{S}^H(q,p-1)}{\hat{\hat{\lambda}}_j} \quad , \quad s = 1,\ldots,p-1.$$

Now assume that p is a power of 2 which implies

$$\mu_s = \frac{2}{p}\sum_{q=1}^{p-1} \frac{\sin\left(\frac{sq\pi}{p}\right)\sin\left(\frac{(p-1)q\pi}{p}\right)}{\hat{\lambda} + 2\hat{\delta}\cos\left(\frac{q\pi}{p}\right)} = \Phi_{s(p-1)}(\hat{\lambda},\hat{\delta},p-1) = -\frac{z_\mu}{\hat{\delta}}\frac{(z_\mu^{-s} - z_\mu^s)}{1-z_\mu^{2p}},$$

where z_μ is the root to $z^2 + \frac{\hat{\lambda}}{\hat{\delta}}z + 1$ with $|z_\mu| < 1$.

So the preconditioned system is defined by

$$
\hat{C}^{-1}C = \begin{pmatrix} 1 & & & \hat{\lambda}'\mu_1 \\ & \ddots & & \vdots \\ & & 1 & \hat{\lambda}'\mu_{p-2} \\ & & & 1+\hat{\lambda}'\mu_{p-1} \end{pmatrix},
$$

which gives that $p-2$ eigenvalues are identically 1 and one equals $1+\hat{\lambda}'\mu_{p-1}$.

Theorem 7.6 *Assume that Δt is fixed. Then*

$$
\lim_{n\to\infty}(1+\hat{\lambda}'\mu_{p-1}) = \frac{1}{p} \tag{49}
$$

Proof:
By observing that

$$
\lim_{n\to\infty} z_{\hat{C}} = -i,
$$

we get

$$
\lim_{n\to\infty}(\hat{\lambda}'\mu_{p-1}) = -\frac{1}{2(n+1)}\cdot\frac{2(n+1)(p-1)}{p},
$$

which gives (49). □

Hence, the only eigenvalue that is different from one, is bounded by $1/p$. Since the dimension of the nullspace of $\hat{C}^{-1}C'$ is $p-2$, the eigenvalue 1 to $\hat{C}^{-1}C$ has $p-2$ linearly independent eigenvectors, and we have convergence in only 2 iterations. From this result we see that the preconditioner obatined from a Toeplitz approximation clusters the eigenvalues extremely well in 1D. We expect that this result in some sense extends to 2D, yielding highly clustered spectra.

7.1.3 The subdomain systems

From [12] we can draw the following conclusions for the preconditioned subdomain systems. PGCR on (7) with preconditioner

- $\hat{A}_{11}^{(q)}(\bullet, D)$, $q = 1,\ldots,p-1$, converges in 1 iteration since $\hat{A}_{11}^{(q)}(\bullet, D) = A_{11}^{(q)}$.
- $\hat{A}_{11}^{(p)}(\bullet, D)$ yields an asymptotic convergence factor ρ defined in (10), which fulfills

  $$
  \rho \le \frac{\sqrt{2+3\phi^2-2\sqrt{1-\phi^2}}}{2},
  $$

  where ϕ is defined in (39) and the time-step Δt is given by

  $$
  \Delta t = ch_1^{\alpha} \quad , \quad 0<\alpha<1 \quad , \quad c>0.
  $$

  The proof of this result can be found in [12].

7.2 PMM

Here we derive upper bounds to the number of iterations needed for PGCR to converge on (3).

Theorem 7.7 *PGCR on (3) converges in at most η iterations, where η can be found in Table 3 for different preconditioners.*

Preconditioner	η
$\hat{\mathcal{A}}(D, \bullet)$	$n(p-1)+m-p+1$
$\hat{\mathcal{A}}(\bullet, D)$	np
$\hat{\mathcal{A}}(D, D)$	$m+n-1$

Table 3: Number of iterations needed for convergence.

Proof:
The proof is mainly the same as the one presented in [11]. It is based on determining $\operatorname{rank}(\hat{\mathcal{A}}^{-1}\mathcal{E})$, where $\mathcal{E} = \mathcal{A} - \hat{\mathcal{A}}$. □

Note that the only preconditioner that yields a η, that is independent of the number of subdomains, is $\hat{\mathcal{A}}(D, D)$. The other preconditioners render an upper bound to the number of iterations, that grows linearly with p. Note also that the PMM-iteration with preconditioner $\hat{\mathcal{A}}(D, D)$ is equivalent to the iteration on (14) with a block-Toeplitz preconditioner defined in [11].

8 Spectra

In this section we present the spectra of both the preconditioned system and the original problem. In each subsection we present both the Schur-Complement system and the Partitioned Matrix system. In all figures the parameters are $n = 23$, $m = 23$, $p = 4$, $m_q = 5$, $q = 1, \ldots, 4$, $\kappa_1 = \kappa_2 = 100$, and $\sigma_1(x_1) = \sigma_2(x_2) = 1$. We also present different properties of the spectra in each case. For that reason we define the *spectral quotient* μ as a measure of the condition of the spectrum

$$\mu = \frac{\max_i |\lambda_i|}{\min_i |\lambda_i|}.$$

Here λ_i are the eigenvalues of the coefficient-matrix considered.

8.1 SCM, the Schur-Complement system

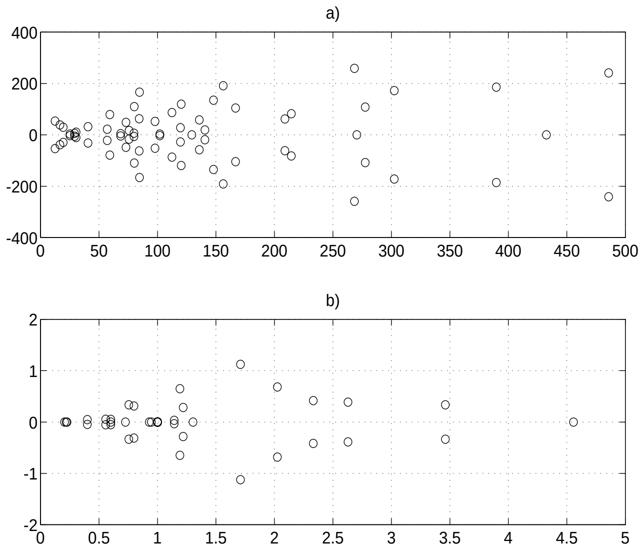

Figure 5: Spectra of the Schur-Complement system with preconditioner a) I, and b) $\hat{C}(D)$.

	Preconditioner	
	None	$\hat{C}(D)$
$\max_i \Re e(\lambda_i)$	484.69	4.56
$\min_i \Re e(\lambda_i)$	12.43	0.21
$\max_i \Im m(\lambda_i)$	258.74	1.12
$\max_i \lvert\lambda_i\rvert$	542.25	4.56
$\min_i \lvert\lambda_i\rvert$	25.38	0.21
μ	21.36	22.16
$\#(\lambda_i{=}1)$	0	34

Table 4: Properties of the spectra of the Schur-Complement system.

In Figure 5 and Table 4 we see that the Schur-Complement system seems to be relatively well-conditioned even without preconditioning. The Toeplitz approximations lead to a preconditioner that yields a well-clustered spectrum.

8.2 SCM, subdomain systems

We only consider the subdomains Ω_q, $q = 1, \ldots, p-1$, since the spectra for Ω_p are shown in [15] and [11].

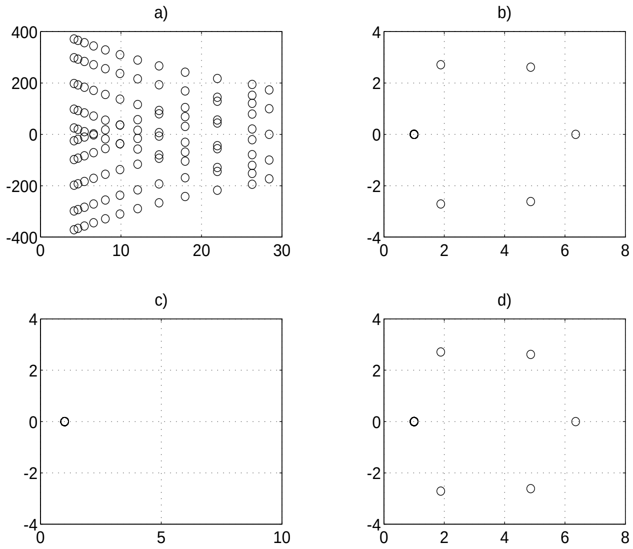

Figure 6: Spectra of a subdomain system with preconditioner a) I, b) $\hat{A}_{11}^{(q)}(D,\bullet)$, c) $\hat{A}_{11}^{(q)}(\bullet,D)$, and d) $\hat{A}_{11}^{(q)}(D,D)$.

	Preconditioner			
	None	$\hat{A}_{11}^{(q)}(D,\bullet)$	$\hat{A}_{11}^{(q)}(\bullet,D)$	$\hat{A}_{11}^{(q)}(D,D)$
$\max_i \Re e(\lambda_i)$	28.41	6.35	1.00	6.35
$\min_i \Re e(\lambda_i)$	4.16	1.00	1.00	1.00
$\max_i \Im m(\lambda_i)$	371.35	2.71	0.00	2.71
$\max_i \lvert\lambda_i\rvert$	371.38	6.35	1.00	6.35
$\min_i \lvert\lambda_i\rvert$	6.93	1.00	1.00	1.00
μ	53.62	6.35	1.00	6.35
$\#(\lambda_i\!=\!1)$	0	110	115	110

Table 5: Properties of the spectra of a subdomain system.

From Figure 6, and Table 5 we immediately see that $\hat{A}_{11}^{(q)}(\bullet, D)$ yields the exact solver and hence we have convergence in 1 iteration which is also pointed out in Section 7.2. The other preconditioners yield highly clustered and well-conditioned systems.

8.3 PMM

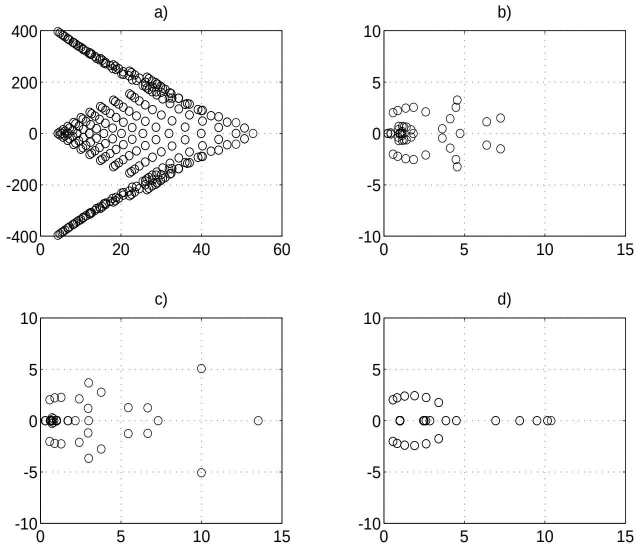

Figure 7: Spectra of the Partitioned Matrix with preconditioner a) I, b) $\hat{\mathcal{A}}(D,\bullet)$, c) $\hat{\mathcal{A}}(\bullet,D)$, and d) $\hat{\mathcal{A}}(D,D)$.

	Preconditioner			
	None	$\hat{\mathcal{A}}(D,\bullet)$	$\hat{\mathcal{A}}(\bullet,D)$	$\hat{\mathcal{A}}(D,D)$
$\max_i \Re e(\lambda_i)$	52.82	7.25	13.53	10.39
$\min_i \Re e(\lambda_i)$	4.32	0.23	0.28	0.57
$\max_i \Im m(\lambda_i)$	396.30	3.24	5.07	2.43
$\max_i \lvert\lambda_i\rvert$	396.32	7.40	13.53	10.39
$\min_i \lvert\lambda_i\rvert$	4.32	0.23	0.28	1.00
μ	91.67	31.62	48.70	10.39
$\#(\lambda_i\!=\!1)$	0	484	488	484

Table 6: Properties of the spectra of the Partitioned Matrix.

Here we see that all the Toeplitz approximations yield highly clustered spectra that are well-conditioned.

9 Results from the implementation on a parallel computer

Here we show the results obtained from a parallel implementation. Both results from an Intel iPSC/2 and an Intel Paragon are presented. The program was originally written for the iPSC/2 at the Department of Scientific Computing, Uppsala University. It was then transferred to the Paragon, located at Para//ab at the University of Bergen.

In all cases we consider PGCR(6), a longer restarting length cannot be motivated due to memory restrictions. However, numerical experiments indicate that the number of iterations decreases considerably with increasing restarting length. As convergence-criterion for a general system of equations (8), we use

$$\frac{||\hat{B}^{-1}(y - Bx_k)||_2}{||\hat{B}^{-1}y||_2} < 10^{-6}.$$

For all iterations we have taken the right-hand side as the initial guess to the solution.

The coefficients σ_1 and σ_2 are chosen as

$$\sigma_d(x_d) = \cosh^2\left(s_d(2x_d - 1)\right)\frac{\tanh(s_d)}{s_d}. \tag{50}$$

This is suitable when we want to simulate stretchings in the physical grid. We then employ the coefficients (50), and perform the calculations in the equidistant computational grid. The scalar parameter s_d determines how much the grid is to be stretched in the x_d-direction. We consider both stretching in one space-dimension, and stretchings in both space-dimensions. Stretching in one space-dimension is interesting for instance when we want to compute a two-dimensional viscous channel flow. In order to resolve the boundary layers, the grid-lines are denser near the solid walls.

In all cases we consider $p = \#subdomains = \#processors$.

9.1 Definitions

To determine how efficient a parallel program is, two quantities have been defined: *Speed up* $S_{p,t}$ and *Rate of efficiency* $E_{p,t}$ and $E_{p,s}$. The speed-up shows how much faster the program executes on p nodes than on one node

$$S_{p,t} = \frac{T_1}{T_p}.$$

Here T_p denotes the execution time on p nodes. T_p could be determined in two different ways. To get the "non-scaled" speed up, T_p is measured as the time required on p nodes when the problem in all is as big as the one on one node. To get the "scaled" speed up, the execution time is measured when each node deals with a problem as big as the one executed on one node. Then the time obtained is divided by p to get T_p.

The rate of efficiency indicates how efficient the program is. Two quantities are here defined, the rate of efficiency in time

$$E_{p,t} = \frac{S_{p,t}}{p},$$

and the rate of efficiency in arithmetic speed

$$E_{p,s} = \frac{M_p}{M_1 p}.$$

Here M_p is the arithmetic speed in floating point operations per second on p processors.

9.2 The method

In this section we study the how the methods depend on different parameters.

9.2.1 SCM

We start by considering the inner iterations discussed in Section 2.2, i.e. the subdomain solves in the matrix-vector multiplications with the Schur-Complement C. The most time-consuming part in the SCM-iteration is the local preconditioner solves. It requires $\sum_{k=1}^{it_{outer}} 2\max_q it_{inner,k}^{(q)}$ such solves to solve (4), where $it_{inner,k}^{(q)}$ is the number of inner iterations required in outer iteration k in Ω_q. For constant coefficients, we see from Section 7.1.3 that most of the time in the iteration will be spent by performing inner iterations in Ω_p. This is due to the fact that $\hat{A}_{11}^{(q)}(\bullet, D) = A_{11}^{(q)}$, $q = 1, \ldots, p-1$, so we get the exact solution from one iteration for these systems. Hence, processor number p will be working most of the time while the others remain idle. This is not a good load-balancing in a parallel method. So, no matter how much we can reduce the number of outer iterations by preconditioning the Schur-Complement system, it will still be the inner iterations in Ω_p that limits the performance of the method. A remedy to this problem is to only consider the grid-line $j = m$ as subdomain Ω_p. Then the coefficient-matrix $A_{11}^{(p)}$ is tridiagonal, and this system can be solved with a direct method using $\mathcal{O}(8n)$ operations. Note also that for $q = 1, \ldots, p-1$ we solve (23) instead of (7) which leads to a reduction in memory requirements. We start by presenting the results from iterating on (4).

The parameters used are $p = 4$, $\kappa_1 = 100$, $s_1 = s_2 = 0$, $m = 49$, $m_q = 15$, $q = 1, \ldots, p-1$, and $m_p = 1$. We present the results for varying ϕ defined in (39).

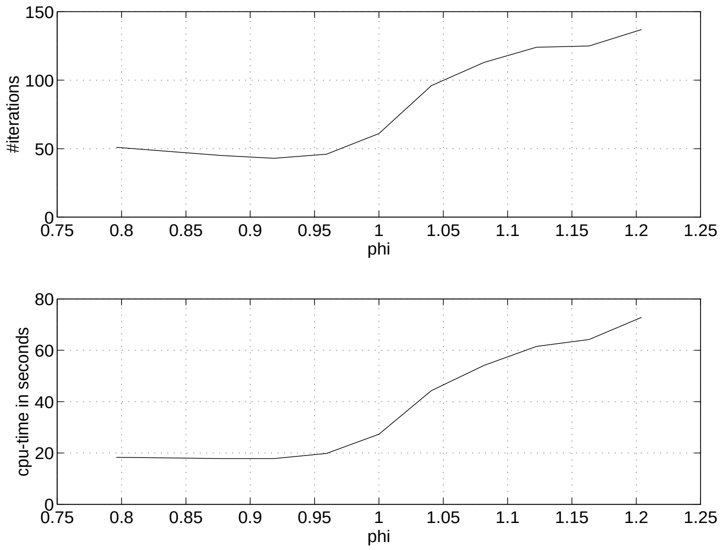

Figure 8: Number of iterations and cpu-time obtained for convergence.

In Figure 8 we see that the iteration-count is *strongly* dependent on ϕ, i.e., the grid ratio. By decreasing ϕ from 1 to $\simeq$ 0.90-0.95 we decrease the number of iterations considerably. Next we show the results from iterating on (4) for different problem-sizes. The parameters used are $p = 4$, $\kappa_1 = 100$, $s_1 = s_2 = 0$, $m_q = (m-p)/(p-1)$, $q = 1, \ldots, p-1$, and $m_p = 1$. Since the grid ratio is so crucial for the performance, we have used $n \approx 0.95 \cdot m$.

n	m	# iterations	cpu-time[s]
24	25	77	8.42
47	49	46	19.8
92	97	27	48.5
183	193	19	146

Table 7: Number of iterations and cpu-time obtained for convergence on an Intel iPSC/2.

From Figure 8 and Table 7 we draw the following conclusions:

- The convergence depends *only* on the grid ratio and *not* on the number of unknowns.

- The way by using only the grid-line $j = m$ as Ω_p is very suitable for constant coefficients.

- The fact that the number of iterations decreases with increasing problem-size, is even more pronounced here compared to [14], [11].

Henceforth we will only consider this nonuniform domain decomposition. The subdomain solves are accomplished through solving (23) with $\hat{A}_{11}(\bullet, D)$ for $s_2 = 0$. For $s_2 > 0$ we solve (7) with an iterative method and preconditioner $\hat{A}_{11}(\bullet, D)$.

Now that we have an efficient SCM-iteration due to the nonuniform decomposition of the domain, we will study the influence of the preconditioner to the Schur-Complement system. The parameters used are $p = 4$, $\kappa_1 = 100$, $m_q = 2^{\log_2(n+1)-1}$, $q = 1, \ldots, p-1$, and $m_p = 1$.

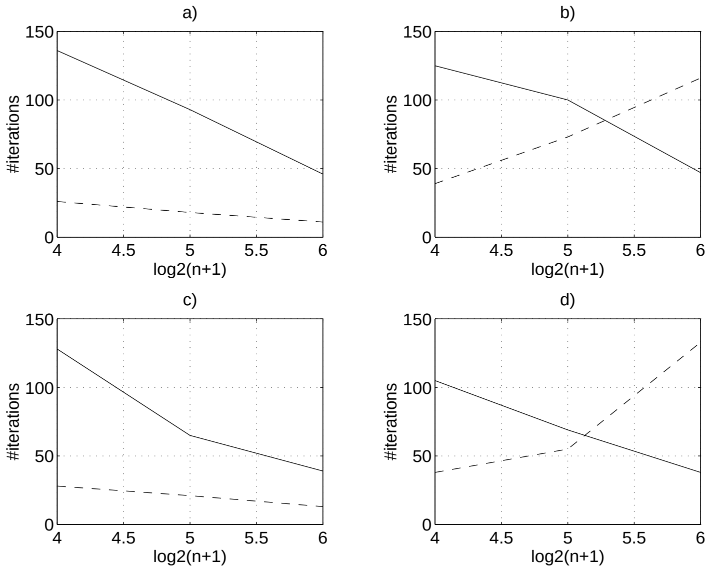

Figure 9: Number of iterations obtained for convergence for a) $s_1 = s_2 = 0.0$ b) $s_1 = 0.5$, $s_2 = 0.0$ c) $s_1 = 0.0$, $s_2 = 0.5$ and d) $s_1 = s_2 = 0.5$. Solid line represents the original system and dashed line the preconditioned system.

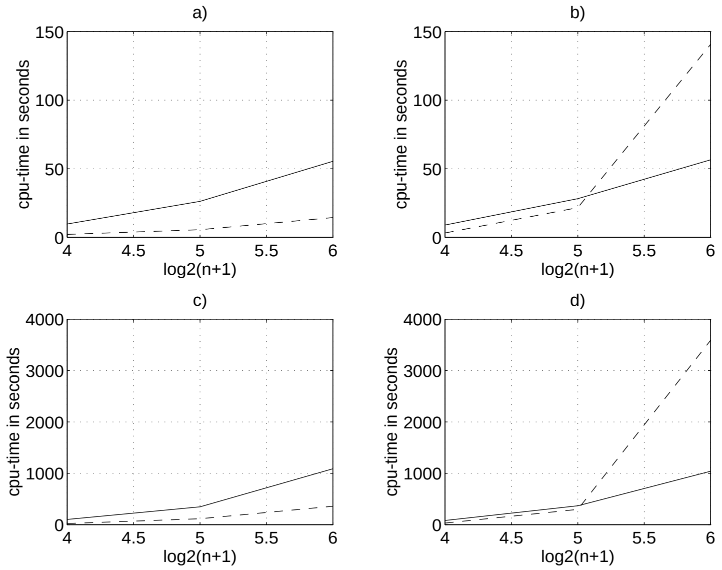

Figure 10: Cpu-time in seconds obtained for convergence for a) $s_1 = s_2 = 0.0$ b) $s_1 = 0.5$, $s_2 = 0.0$ c) $s_1 = 0.0$, $s_2 = 0.5$ and d) $s_1 = s_2 = 0.5$. Solid line represents the original system and dashed line the preconditioned system. The timings are from an Intel iPSC/2.

From Figure 9 and 10 we draw the following conclusions:

- For $s_1 = 0$ the preconditioner reduces both the number of iterations and the cpu-time.
- For $s_1 = 0.5$ the preconditioner works good for small problems, but for larger problems, the iteration on the original system is preferable.
- The iteration on the original system is independent of the stretching in the grid. Note however that in all four cases, the stretching is quite moderate.
- For $s_2 = 0$ we can solve (23) instead of (7) which reduces the cpu-time considerably.

Now we have found out how the SCM-method depends on the problem-size and the stretching in the grid. The next thing we will study is the dependency on the time-step, i.e. we vary κ_1 and κ_2. In Figure 11 we study the behavior of the original system and in Figure 12 we consider the preconditioned system. Here we have used $p = 4$. In Figure 11 $n \approx 0.9 \cdot m$ and in Figure 12 we have $n + 1 = 2 \cdot ((m - p)/(p - 1) + 1)$.

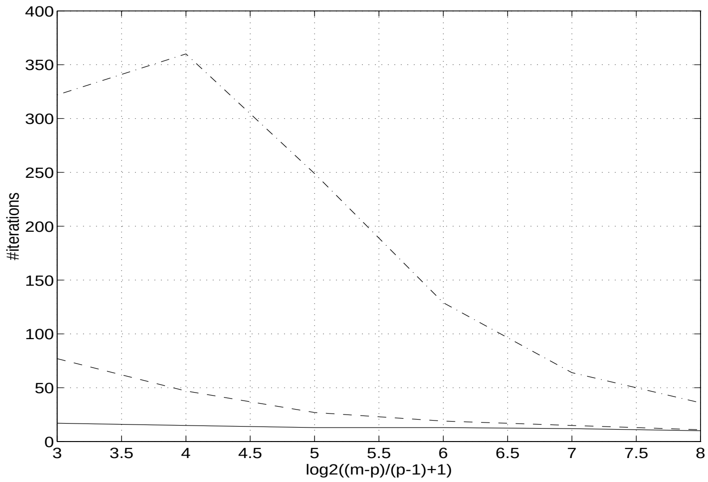

Figure 11: Number of iterations for $\kappa_1 = 10$ solid line, $\kappa_1 = 100$ dashed line, and $\kappa_1 = 1000$ dashed-dotted line.

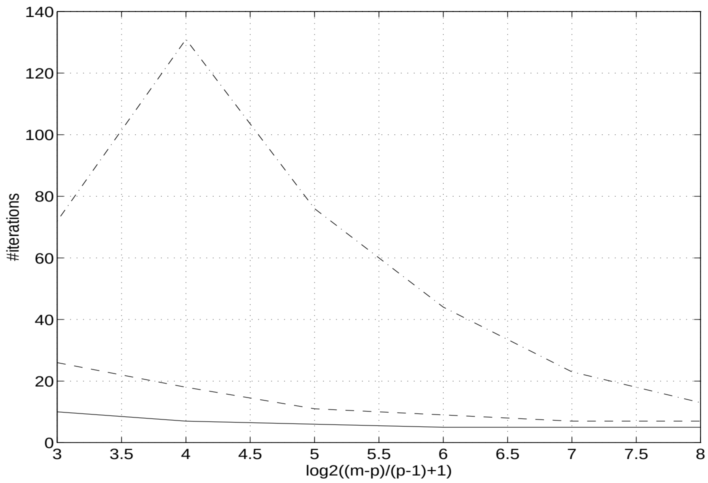

Figure 12: Number of iterations for $\kappa_1 = 10$ solid line, $\kappa_1 = 100$ dashed line, and $\kappa_1 = 1000$ dashed-dotted line.

In Figures 11 and 12 we see that the number of iterations decreases with increasing problem-size with very few deviations. We also see that the iteration-count increases with increasing time-step, but the increase is less for larger problem-sizes. Note that the shapes of the curves from the original system and the preconditioned system are similar.

Finally we have examined the behavior when we increase the number of subdomains for the same problem, i.e. the the scalability of the method. Note that we have to vary the problem-size slightly when we vary the number of subdomains due to the fast modified sine transforms. We have used the parameters $\kappa_1 = 100$ and $\phi = 0.9$.

$\hat{C}$	n	m	p	# iterations	cpu-time[s]
I	63	97	4	46	55.3
I	63	113	8	281	158
I	63	121	16	940	255
$\hat{C}(D)$	63	97	4	11	14.1
$\hat{C}(D)$	63	113	8	15	9.47
$\hat{C}(D)$	63	121	16	24	7.79

Table 8: Number of iterations and cpu-time obtained for convergence on an Intel iPSC/2.

As seen in Table 8 the iteration on the original system clearly isn't very scalable. The preconditioned system on the other hand seems to be quite scalabe. We shall further investigate this in Section 9.3.

9.2.2 PMM

Now we consider the PMM-iteration (3). We start by running on 4 processors with $n = m = 63$, $\kappa_1 = 100$ for different stretchings. A -, means that the iteration did not converge within 1500 iterations.

Preconditioner	s_1	s_2	#iterations	cpu-time[s]
$\hat{\mathcal{A}}(D, \bullet)$	0	0	87	64.5
	0.5	0	-	-
	0	0.5	96	71.1
	0.5	0.5	1493	1089
$\hat{\mathcal{A}}(\bullet, D)$	0	0	90	63.1
	0.5	0	127	88.6
	0	0.5	109	76.2
	0.5	0.5	168	117
$\hat{\mathcal{A}}(D, D)$	0	0	62	46.4
	0.5	0	-	-
	0	0.5	104	77.0
	0.5	0.5	-	-

Table 9: Number of iterations and cpu-time obtained for convergence on an Intel iPSC/2.

Here we see that $\hat{\mathcal{A}}(\bullet, D)$ is the best preconditioner to (3). In some cases there are other preconditioners that give convergence in fewer iterations, but they do not converge for all test-cases. They are therefore considered as inferior. In the following we only consider preconditioner $\hat{\mathcal{A}}(\bullet, D)$. Note that we cannot vary ϕ defined in (39) with this preconditioner, due to the fast modified sine transforms in both space-directions.

Next we study the behavior when we vary both problem-size and time-step. In Figure 13 we have used the parameters $p = 4$ and $n = m$.

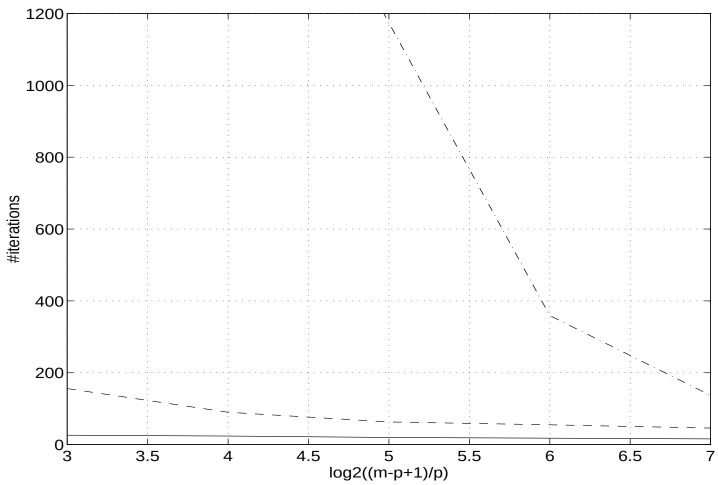

Figure 13: Number of iterations for $\kappa_1 = 10$ solid line, $\kappa_1 = 100$ dashed line, and $\kappa_1 = 1000$ dashed-dotted line.

From Figure 13 we conclude that the number of iterations:

- decreases with increasing problem-size
- increases with increasing time-step but less for larger problem-sizes.

Finally we consider the speed-up properties of the PMM-algorithm, for a non-scaled problem. Here $n = m = 63$, and $\kappa_1 = 10$.

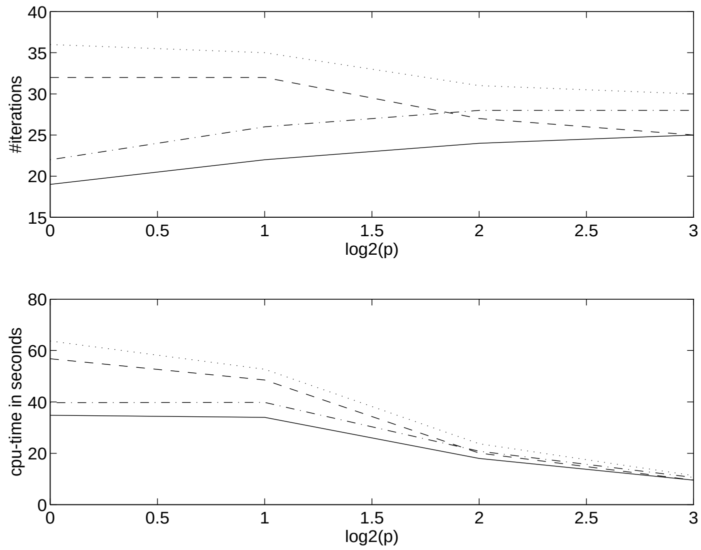

Figure 14: Number of iterations and cpu-time obtained for convergence. Solid line represents $s_1 = s_2 = 0$, dashed line $s_1 = 0$, $s_2 = 0.5$, dashed-dotted line $s_1 = 0.5$, $s_2 = 0$, and dotted line $s_1 = s_2 = 0.5$.

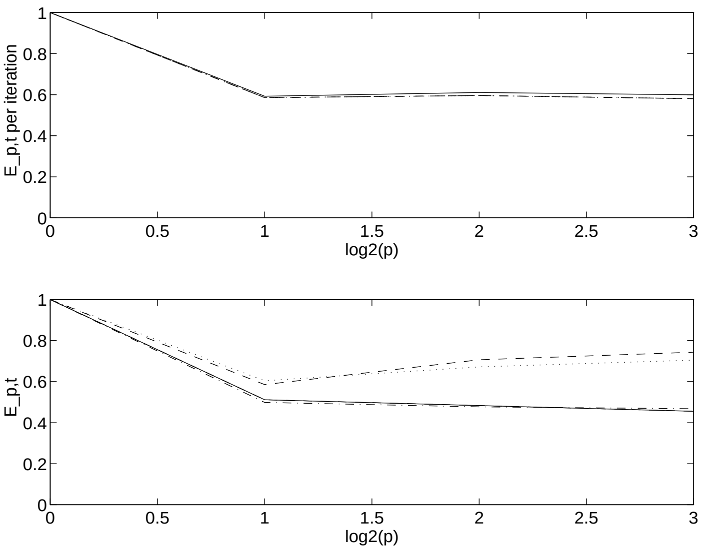

Figure 15: Rate of efficiency, both per iteration, and for the whole iteration. Solid line represents $s_1 = s_2 = 0$, dashed line $s_1 = 0$, $s_2 = 0.5$, dashed-dotted line $s_1 = 0.5$, $s_2 = 0$, and dotted line $s_1 = s_2 = 0.5$.

In Figure 14 we see that for the problems with stretching in the x_2-direction, both the time and the number of iterations decrease with increasing number of subdomains. For the other problems, the number of iterations increase, but still the cpu-time goes down. From Figure 15 we conclude that the rate of efficiency per iteration is the same in all test-cases as expected. Note also that the decrease in the efficiency is between 1 and 2 processors. For 4 and 8 processors, the rate of efficiency stays approximately the same. If we take into account that the number of iterations vary with p, the problems with $s_2 > 0$ get a higher rate of efficiency, while the other problems get a lowering in efficiency compared to the efficiency per iteration.

9.3 Parallel computing

In this section we present the results from an Intel Paragon.

9.3.1 SCM

Here we consider the solution of the whole problem, i.e., **Algorithm SCM**. The Schur-Complement system is preconditioned with preconditioner $\hat{C}(D)$. Here cpu-time$_{it}$ is the time to solve the Schur-Complement system and cpu-time$_{tot}$ is the time to solve the whole problem. # iter. refers to the number of iterations to solve the Schur-Complement system. The parameter used is $\kappa_1 = 100$. To obtain the rate of efficiency, the reference level is set to $p = 4$.

p	n	m	# iter.	cpu-time$_{it}$[s]	cpu-time$_{tot}$[s]	Mflops	$E_{p,t}$	$E_{p,s}$
4	255	385	7	9.77	13.2	9.99	1.00	1.00
8	255	449	9	5.85	7.64	22.6	0.86	1.13
16	255	481	12	3.86	4.97	43.3	0.67	1.08
32	255	497	17	3.07	3.75	74.6	0.44	0.93
64	255	505	26	3.00	3.43	116	0.24	0.73

Table 10: Number of iterations, cpu-time, arithmetic speed and rate of efficiency on an Intel Paragon.

The reason why we obtain a rate of efficiency > 1 for the arithmetic speed is that the inefficiency due to the nonuniform decomposition of the domain becomes less pronounced the more subdomains that we have. Note that the poor rate of efficiency in time partly is due to the fact that we are solving larger problems when we have more processors.

We end this section by presenting the results from some "large" problems run. Again $\kappa_1 = 100$.

p	n	m	# iterations	cpu-time$_{it}$[s]	cpu-time$_{tot}$[s]	Mflops
32	511	993	11	7.12	9.02	92.3
32	1023	993	10	12.5	16.0	96.9
32	1023	1985	9	31.4	24.4	125
64	1023	1009	12	8.60	10.5	163
64	1023	2017	11	14.6	18.3	186
64	2047	2017	9	26.0	36.7	159

Table 11: Number of iterations, cpu-time and arithmetic speed on an Intel Paragon for some "large problems".

9.3.2 PMM

We start by presenting the rate of efficiency, for both a scaled and a non-scaled problem. To obtain the rate of efficiency, we have compared with the timings on 4 processors.

p	n	m	#iter.	cpu-time[s]	Mflops	$E_{p,t}$	$E_{p,t}/it$	$E_{p,s}$
4	255	255	55	32.7	17.8	1.00	1.00	1.00
8	255	255	55	16.4	33.5	1.00	1.00	0.94
16	255	255	56	9.26	57.5	0.88	0.90	0.81
32	255	255	57	5.93	88.0	0.69	0.71	0.62
4	255	255	55	32.7	17.8	1.00	1.00	1.00
8	255	511	22	15.1	30.9	2.17	0.87	0.87
16	255	1023	21	14.9	59.7	2.19	0.84	0.84

Table 12: Number of iterations, cpu-time obtained for convergence, the arithmetic speed, and the rate of efficiency for $\kappa_1 = 100$, and $s_1 = s_2 = 0$.

In Table 12 we see that the rate of efficiency is very high in all test-cases. The extremely high rate of efficiency for the non-scaled problem on 8 processors is probably due to very low rate of caching. Note that $E_{p,t} > 2$ in some cases which is extremely good. This unexpected phenomenom is due to the fact that the number of iterations decreases considerably with increasing problem-sizes.

In Table 13 we present the performance for some "large" problems run.

p	n	m	#iterations	cpu-time[s]	Mflops
32	511	1023	23	16.7	110
32	1023	1023	43	58.8	118
64	1023	1023	45	29.8	231
64	1023	2047	22	32.8	216
64	2047	2047	40	134	192

Table 13: The performance for some "large" problems, for $\kappa_1 = 100$, and $s_1 = s_2 = 0$.

The final result is from a stretched grid, where we have used $s_1 = s_2 = 0.8$. To see the influence of the decomposition of the domain we have run the same problem on different number of processors.

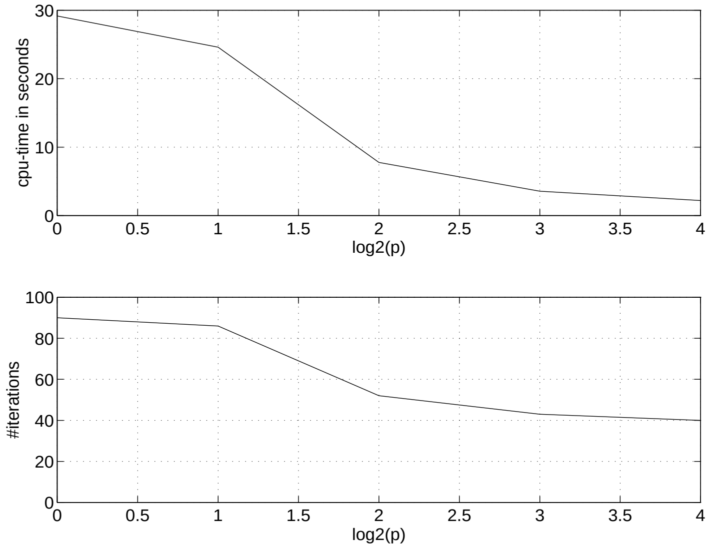

Figure 16: Number of iterations and cpu-time for $n = m = 127$, $\kappa_1 = 10$, $s_1 = s_2 = 0.8$.

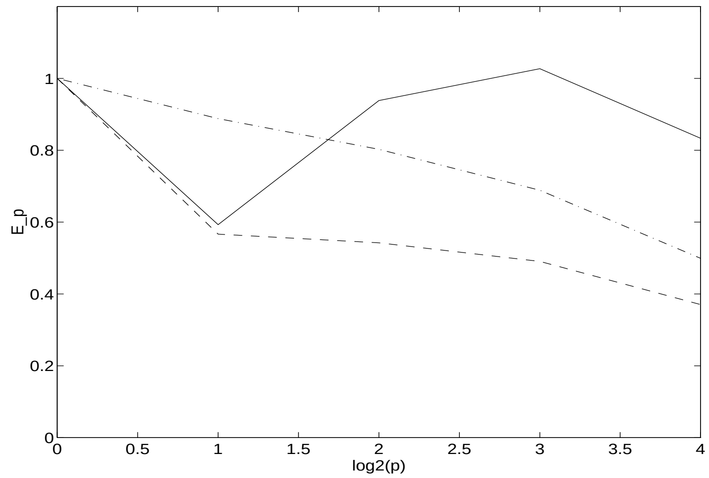

Figure 17: Rate of efficiency per iteration dashed line, for the whole iteration solid line, and for the Mflops dashed-dotted line, for $n = m = 127$, $\kappa_1 = 10$, $s_1 = s_2 = 0.8$.

In Figures 16, and 17 we see again that the number of iterations decreases, when we increase the number of subdomains for a problem with varying coefficients. This is not surprising since we are taking averages over less varying coefficients, when the subdomains get thinner. Hence, we can consider domain decomposition as a remedy to the problems with variable coefficients seen in [15] and [11].

10 Conclusions

In this report we have presented two variants of a domain decomposition method to solve first-order pdes in two space-dimensions. Both a so called Schur-Complement Method and a Partitioned Matrix Method are discussed. The methods are analyzed theoretically and numerical results corroborating the theory have been performed.

We have defined preconditioners built on Toeplitz blocks, which have fast solvers based on a fast modified sine transform. Numerical spectra for the preconditioned systems are presented, showing highly clustered spectra.

Due to a nonuniform decomposition of the domain, the Schur-Complement method was very successful for constant coefficients in the x_2-direction. The preconditioned system was the one that gave the best results. This method was also highly scalable. However, we have shown theoretically and numerically that the Schur-Complement system can be solved successfully without preconditioning, for a decomposition with only few subdomains. Hence, this method can be used for more irregular domains. The PMM-method and the preconditioned SCM-method on the other hand require logically rectangular domains, due to the solver of the preconditioner.

The results from a parallel implementation show that the PMM-method yield few iterations, and that the scalability of the method is very high. Finally we have shown that for problems with variable coefficients, the decomposition of the domain leads to a considerable reduction in number of iterations. This is due to the fact that the variable coefficients are treated as piecewice constants in the preconditioner. For the original problem (14), the variable coefficients are handled by a global average of the coefficients.

11 Acknowledgments

I would like to thank my advisor Professor Bertil Gustafsson for helpful support. I also want to express my gratitude to Doctor Sverker Holmgren and Doctor Kurt Otto for sharing their wide experience of iterative methods and preconditioners for nonsymmetric problems with me. Finally I am very grateful to Para//ab at the University of Bergen for letting me use their Paragon. The system administrator Bjarne Herland gave invaluable advice.

References

[1] O. Axelsson, *A restarted version of a generalized preconditioned preconditioned conjugate gradient method*, Report No. 8710, Department of Mathematics, University of Nijmegen, Nijmegen, the Netherlands, 1987.

[2] O. Axelsson and G. Lindskog, *On the eigenvalue distribution of a class of preconditioning methods*, Numer. Math., 48 (1986), pp. 479-498.

[3] Petter Björstad and Olov Widlund, *Iterative Methods for the Solution of Elliptic Problems on Regions Partitioned into Substructures*, SIAM J. Numer. Anal., 23/6 December (1986), pp. 1097-1120.

[4] Tony F. Chan, *Analysis of Preconditioners for Domain Decomposition*, SIAM J. Numer. Anal., 24/2 April (1987), pp. 382-390.

[5] M. Dryja, *A Capacitance Matrix Method for Dirichlet Problem on Polygonal Region*, Numer. Math., 39 (1982), pp. 51-64.

[6] H. C. Elman, *Iterative methods for large sparse nonsymmetric systems of linear equations*, Ph. D. thesis and Report RR-229, Departments of Computer Science, Yale University, New Haven, CT, 1982.

[7] R. W. Freund, G. H. Golub, and N. M. Nachtigal, *Iterative solution of linear systems*, Acta Numerica (1992), pp. 57-100.

[8] J. Guerra and B. Gustafsson, *A semi-implicit method for hyperbolic problems with different time-scales*, SIAM J. Numer. Anal., 23 (1986), pp. 734-749.

[9] B. Gustafsson and H. Stoor, *Navier-Stokes equations for almost incompressible flow*, SIAM J. Numer. Anal., 28 (1991), pp. 1523-1547. 122, Department of Scientific Computing, Uppsala University, Uppsala, Sweden, 1989.

[10] L. Hemmingsson, *A Fast Modified Sine Transform for Solving Block-Tridiagonal Systems with Toeplitz Blocks.*, To appear in Numer. Algor.

[11] L. Hemmingsson, *Toeplitz Preconditioners with Block-structure for First-order PDEs*, Submitted to Numer. Lin. Alg.

[12] L. Hemmingsson and K. Otto, *Analysis of Semi-Toeplitz Preconditioners for First-orde PDEs*, Submitted to SIAM J. Sci. Comp.

[13] R.W. Hockney and C.R. Jesshope, *Parallel computers 2*, IOP Publishing Ltd, 1988.

[14] S. Holmgren and K. Otto, *A comparison of preconditioned iterative methods for nonsymmetric block-tridiagonal systems of equations*, Submitted to Appl. Num. Math.

[15] S. Holmgren and K. Otto, *Iterative solution methods and preconditioners for block-tridiagonal systems of equations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp.863-886.

[16] David E. Keyes and William D. Gropp, *A Comparison of Domain Decomposition Techniques for Elliptic Partial Differential Equations and their Parallel Implementation*, SIAM J. Sci. Stat. Comput. 8/7 March (1987), pp. 166-202.

[17] K. Otto, *Analysis of preconditioners for hyperbolic PDE*, Submitted to SIAM J. Numer. Anal.

[18] Y. Saad, *Krylov subspace methods for solving large unsymmetric linear systems*, Math. Comp., 37 (1981), pp. 105-126.

[19] H. A. Schwartz, *Gesammelte mathematische Abhandlungen*, Springer, Berlin, 2, 1890, pp. 133-134.

[20] H. Stoor, *Numerical Solution of the Navier-Stokes equations for small Mach numbers*, Ph.D. thesis, Department of Scientific Computing, Uppsala University, Uppsala, Sweden, 1990.

[21] H.A. van der Vorst, *Preconditioning by incomplete decompositions*, Ph.D. thesis, Rijksuniversiteit Utrecht, Utrecht, the Netherlands, 1982.